E E S

Engineering Equation Solver

for Microsoft Windows
Operating Systems

F-Chart Software
4406 Fox Bluff Rd
Middleton, WI 53562
Phone: 608-836-8531
FAX: 608-836-8536
http://www.fchart.com

The authors make no guarantee that the program is free from errors or that the results produced with it will be free of errors and assume no responsibility or liability for the accuracy of the program or for the results which may come from its use.

EES was compiled with DELPHI by Borland

Registration Number___________________________

ALL CORRESPONDENCE MUST INCLUDE THE REGISTRATION NUMBER

v4/7.01

E E S

Engineering Equation Solver

for Microsoft Windows
Operating Systems

F-Chart Software
4406 Fox Bluff Rd
Middleton, WI 53562
Phone: 608-836-8531
FAX: 608-836-8536
http://www.fchart.com

Table of Contents

Overview

EES (pronounced 'ease') is an acronym for Engineering Equation Solver. The basic function provided by EES is the solution of a set of algebraic equations. EES can also solve differential equations, equations with complex variables, do optimization, provide linear and non-linear regression and generate publication-quality plots. Versions of EES have been developed for Apple Macintosh computers and for the Windows operating systems. This manual describes the version of EES developed for Microsoft Windows operating systems, including Windows 3.1, Windows 95, and Windows NT.

There are two major differences between EES and existing numerical equation-solving programs. First, EES automatically identifies and groups equations which must be solved simultaneously. This feature simplifies the process for the user and ensures that the solver will always operate at optimum efficiency. Second, EES provides many built-in mathematical and thermophysical property functions useful for engineering calculations. For example, the steam tables are implemented such that any thermodynamic property can be obtained from a built-in function call in terms of any two other properties. Similar capability is provided for most organic refrigerants (including some of the new blends), ammonia, methane, carbon dioxide and many other fluids. Air tables are built-in, as are psychrometric functions and JANAF table data for many common gases. Transport properties are also provided for most of these substances.

The library of mathematical and thermophysical property functions in EES is extensive, but it is not possible to anticipate every user's need. EES allows the user to enter his or her own functional relationships in three ways. First, a facility for entering and interpolating tabular data is provided so that tabular data can be directly used in the solution of the equation set. Second, the EES language supports user-written functions and procedure similar to those in Pascal and FORTRAN. EES also provides support for user-written modules, which are self-contained EES programs that can be accessed by other EES programs. The functions, procedures, and modules can be saved as library files which are automatically read in when EES is started. Third, compiled functions and procedures, written in a high-level language such as Pascal, C or FORTRAN, can be dynamically-linked into EES using the dynamic link library capability incorporated into the Windows operating system. These three methods of adding functional relationships provide very powerful means of extending the capabilities of EES.

The motivation for EES rose out of experience in teaching mechanical engineering thermodynamics and heat transfer. To learn the material in these courses, it is necessary for the student to work problems. However, much of the time and effort required to solve problems results from looking up property information and solving the appropriate equations. Once the student is familiar with the use of property tables, further use of the tables does not contribute to the student's grasp of the subject; nor does algebra. The time and effort required to do problems in the conventional manner may actually detract from learning of the subject matter by forcing the student to be concerned with the order in which the equations should be solved (which really does not matter) and by making parametric studies too laborious. Interesting practical problems that may have implicit solutions, such as those involving both thermodynamic and heat transfer considerations, are often not assigned because of their mathematical complexity. EES allows the user to concentrate more on design by freeing him or her from mundane chores.

EES is particularly useful for design problems in which the effects of one or more parameters need to be determined. The program provides this capability with its Parametric Table, which is similar to a spreadsheet. The user identifies the variables which are independent by entering their values in the table cells. EES will calculate the values of the dependent variables in the table. The relationship of the variables in the table can then be displayed in publication-quality plots. EES also provides capability to propagate the uncertainty of experimental data to provide uncertainty estimates of calculated variables. With EES, it is no more difficult to do design problems than it is to solve a problem for a fixed set of independent variables.

EES offers the advantages of a simple set of intuitive commands which a novice can quickly learn to use for solving any algebraic problems. However, the capabilities of this program are extensive and useful to an expert as well. The large data bank of thermodynamic and transport properties built into EES is helpful in solving problems in thermodynamics, fluid mechanics, and heat transfer. EES can be used for many engineering applications; it is ideally suited for instruction in mechanical engineering courses and for the practicing engineer faced with the need for solving practical problems.

The remainder of this manual is organized into seven chapters and five appendices. A new user should read Chapter 1 which illustrates the solution of a simple problem from start to finish. Chapter 2 provides specific information on the various functions and controls in each of the EES windows. Chapter 3 is a reference section that provides detailed information for each menu command. Chapter 4 describes the built-in mathematical and thermophysical property functions and the use of the Lookup Table for entering tabular data. Chapter 5 provides instructions for writing EES functions, procedures and modules and saving them in Library files. Chapter 6 describes how compiled functions and procedures, written as

Windows dynamic-link library (DLL) routines, can be integrated with EES. Chapter 7 describes a number of advanced features in EES such as the use of string, complex and array variables, the solution of simultaneous differential and algebraic equations, and property plots. Appendix A contains a short list of suggestions. Appendix B describes the numerical methods used by EES. Appendix C shows how additional property data may be incorporated into EES. A number of example problems are provided in the Examples subdirectory included with EES. Appendix D indicates which features are illustrated in the example problems provided with EES.

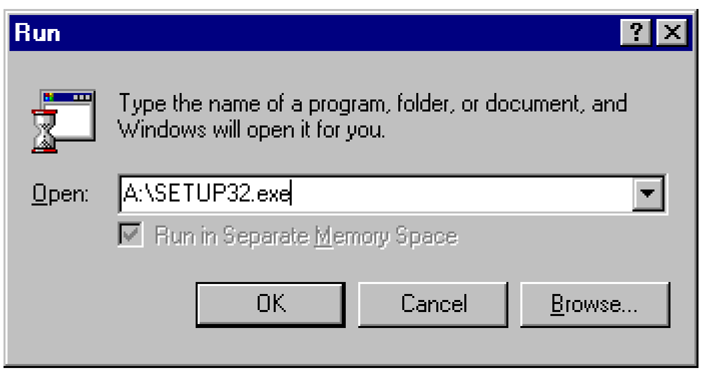
CHAPTER 1

Getting Started

Installing EES on your Computer

There are two versions of EES: EES and EES32. EES is designed to operate with any of the Microsoft Windows operating systems. EES32 is a 32-bit version of the program that will operate only under Windows 95 and NT. This manual is applicable to both versions. However some of the newer features of the program, e.g., modules (Chapter 5) and complex variables (Chapter 7) are implemented only in the 32-bit version. If you are using Windows 95 or NT, you should install the 32-bit version of EES.

EES is distributed in a self-installing compressed form in a file called SETUP16.exe (16-bit version) or SETUP32.exe (32-bit version). To install EES or EES32, execute the installation program. In Windows 95 or NT, the installation program can be executed by selecting the Run command from the Start menu and entering A:\SETUP16.exe or A:\SETUP32.exe.



Here A: is your floppy drive designation. The installation program will provide a series of prompts which will lead you through the complete installation of the EES program.

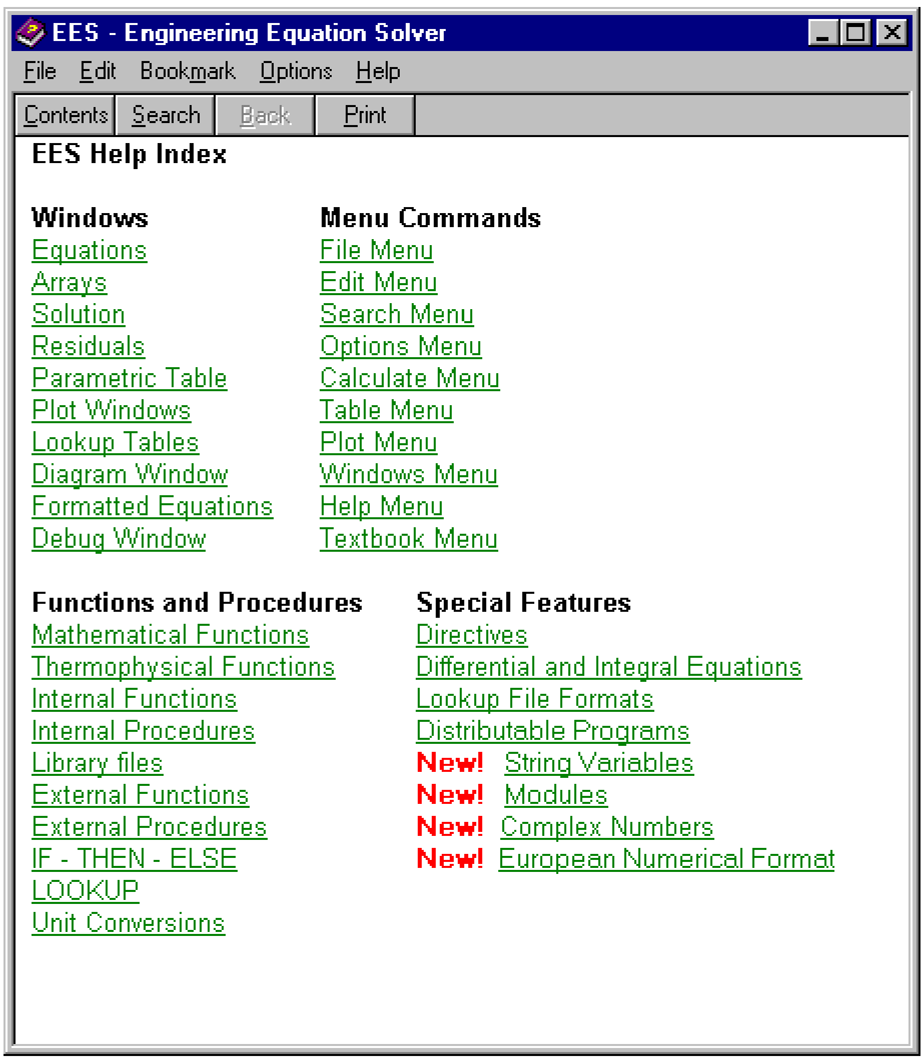
Starting EES 

The default installation program will create a directory named \EESW (16-bit version) or \EES32 (32-bit version) in which the EES files are placed. The EES program icon shown above will identify both the program and EES files. Double-clicking the left mouse button on the EES program or file icon will start the program. If you double-clicked on an EES file, that file will be automatically loaded. Otherwise, EES will load the HELLO.EES file which briefly describes the new features in your version. You can delete or rename the HELLO.EES file if you do not wish to have it appear when the program is started.

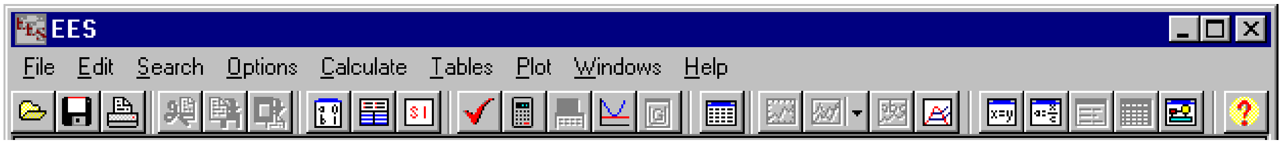
Background Information

EES begins by displaying a dialog window which shows registration information, the version number and other information. The version number and registration information will be needed if you request technical support. Click the OK button to dismiss the dialog window.

Detailed help is available at any point in EES. Pressing the F1 key will bring up a Help window relating to the foremost window. Clicking the Contents button will present the Help index shown below. Clicking on an underlined word (shown in green on color monitors) will provide help relating to that subject.



EES commands are distributed among nine pull-down menus. (A tenth user-defined menu can be placed to the right of the Help menu. See the discussion of the Load Textbook command File menu in Chapter 3.) A brief summary of their functions follows. Detailed descriptions of the commands appear in Chapter 3.



Note the a toolbar is provided below the menu bar. The toolbar contains small buttons which provide rapid access to many of the most frequently used EES menu commands. If you move the cursor over a button and wait for a few second, a few words will appear to explain the function of that button. The toolbar can be hidden, if you wish, with a control in the Preferences dialog (Options menu).

The System menu represented by the EES icon appears above the file menu. The System menu is not part of EES, but rather a feature of the Windows Operating System. It holds commands which allow window moving, resizing, and switching to other applications.

The File menu provides commands for loading, merging and saving work files and libraries, and printing.

The Edit menu provides the editing commands to cut, copy, and paste information.

The Search menu provides Find and Replace commands for use in the Equations window.

The Options menu provides commands for setting the guess values and bounds of variables, the unit system, default information, and program preferences. A command is also provided for displaying information on built-in and user-supplied functions.

The Calculate menu contains the commands to check, format and solve the equation set.

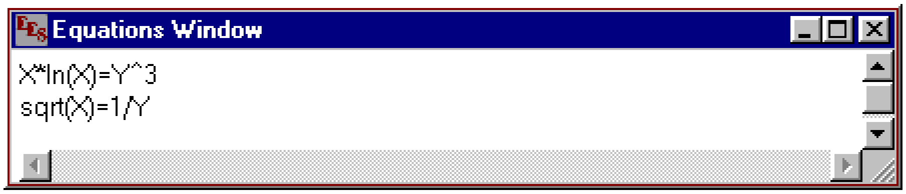
The Tables menu contains commands to set up and alter the contents of the Parametric and Lookup Tables and to do linear regression on the data in these tables. The Parametric Table, similar to a spreadsheet, allows the equation set to be solved repeatedly while varying the values of one or more variables. The Lookup table holds user-supplied data which can be interpolated and used in the solution of the equation set.

The Plot menu provides commands to modify an existing plot or prepare a new plot of data in the Parametric, Lookup, or Array tables. Curve-fitting capability is also provided.

The Windows menu provides a convenient method of bringing any of the EES windows to the front or to organize the windows.

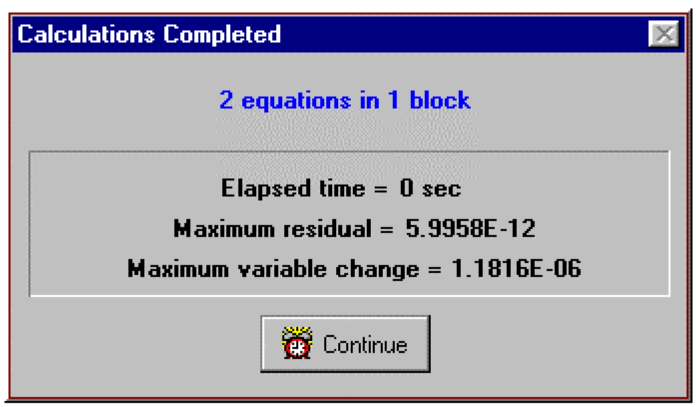
The Help menu provides commands for accessing the online help documentation.

The basic capability provided by EES is the solution of a set of non-linear algebraic equations. To demonstrate this capability, start EES and enter this simple example problem in the Equations window. Note that EES makes no distinction between upper and lower case letters and the ^ sign (or **) is used to signify raising to a power.

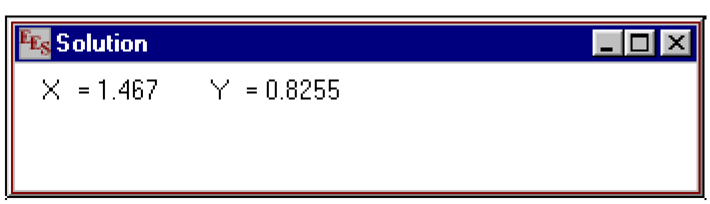


If you wish, you may view the equations in mathematical notation by selecting the Formatted Equations command from the Windows menu.

Select the Solve command from the Calculate menu. A dialog window will appear indicating the progress of the solution. When the calculations are completed, the button changes from Abort to Continue.



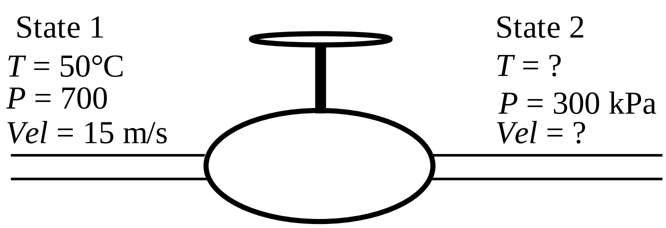
Click the Continue button. The solution to this equation set will then be displayed.



An Example Thermodynamics Problem

A simple thermodynamics problem will be set up and solved in this section to illustrate the property function access and equation solving capability of EES. The problem, typical of that which may be encountered in an undergraduate thermodynamics course, is as follows.

Refrigerant-134a enters a valve at 700 kPa, 50°C with a velocity of 15 m/s. At the exit of the valve, the pressure is 300 kPa. The inlet and outlet fluid areas are both 0.0110 m^2. Determine the temperature, mass flow rate and velocity at the valve exit.



To solve this problem, it is necessary to choose a system and then apply mass and energy balances. The system is the valve. The mass flow is steady, so that the mass balance is:

$$\dot{m}_1 = \dot{m}_2 \tag{1}$$

where

$$\dot{m}_1 = A_1\ Vel_1 / v_1 \tag{2}$$

$$\dot{m}_1 = A_2\ Vel_2 / v_2 \tag{3}$$

$\dot{m}$ = mass flowrate [kg/s]
A = cross-sectional area [m^2]
Vel = velocity [m/s]
v = specific volume [m^3/kg]

We know that

$$A_1 = A_2 \tag{4}$$

The valve is assumed to be well-insulated with no moving parts. The heat and work effects are both zero. A steady-state energy balance on the valve is:

$$\dot{m}_1\left(h_1 + \frac{Vel_1^2}{2}\right) = \dot{m}_2\left(h_2 + \frac{Vel_2^2}{2}\right) \tag{5}$$

where h is the specific enthalpy and $Vel^2/2$ is the specific kinetic energy. In SI units, specific enthalpy normally has units of [kJ/kg] so some units conversions may be needed.

EES provides unit conversion capabilities with the CONVERT function as documented in Chapter 4.

From relationships between the properties of R134a:

$$v_1 = v(T_1, P_1) \quad (6)$$

$$h_1 = h(T_1, P_1) \quad (7)$$

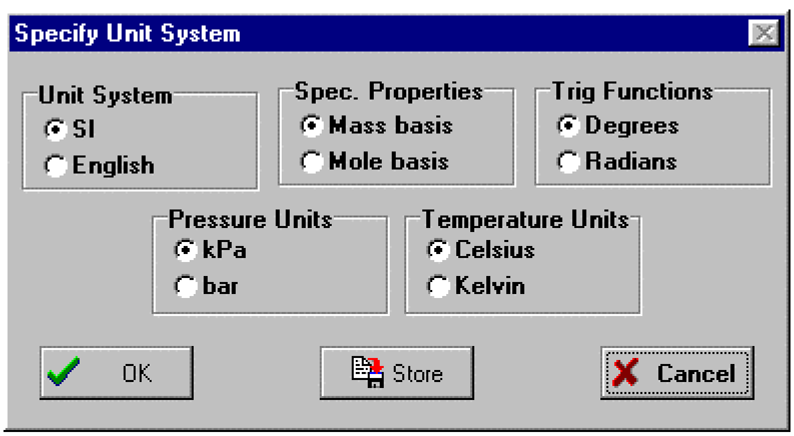
$$v_2 = v(T_2, P_2) \quad (8)$$

$$h_2 = h(T_2, P_2) \quad (9)$$

Ordinarily, the terms containing velocity are neglected, primarily because the kinetic energy effects are usually small and also because these terms make the problem difficult to solve. However, with EES, the computational difficulty is not a factor. The user can solve the problem with the kinetic energy terms and judge their importance.

The values of T_1, P_1, A_1, Vel_{11} and P_2 are known. There are nine unknowns: A_2, $\dot{m}_1$, $\dot{m}_2$, Vel_2, h_1, v_1, h_2, v_2, T_2. Since there are 9 equations, the solution to the problem is defined. It is now only necessary to solve the equations. This is where EES can help.

Start EES and select the New command from the File menu. A blank Equations window will appear. Before entering the equations, however, set the unit system for the built-in thermophysical properties functions. To view or change the unit system, select Unit System from the Options menu.

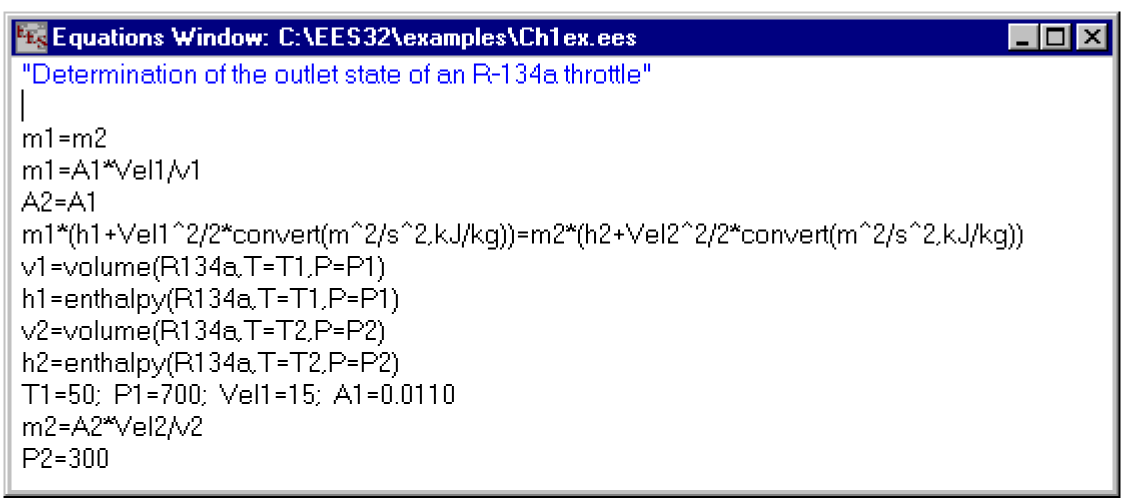


EES is initially configured to be in SI units with T in °C, P in kPa, and specific property values in their customary units on a mass basis. These defaults may have been changed during a previous use. Click on the controls to set the units as shown above. Click the OK button (or press the Return key) to accept the unit system settings.

The equations can now be entered into the Equations window. Text is entered in the same manner as for any word processor. Formatting rules are as follows:

1. Upper and lower case letters are not distinguished. EES will (optionally) change the case of all variables to match the manner in which they first appear.
2. Blank lines and spaces may be entered as desired since they are ignored.
3. Comments must be enclosed within braces { } or within quote marks " ". Comments may span as many lines as needed. Comments within braces may be nested in which case only the outermost set of { } are recognized. Comments within quotes will also be displayed in the Formatted Equations window.
4. Variable names must start with a letter and consist of any keyboard characters except ( ) ‘ | * / + - ^ { } : " or ;. Array variables (Chapter 7) are identified with square braces around the array index or indices, e.g., X[5,3]. String variables (Chapter 7) are identified with a $ as the last character in the variable name. The maximum length of a variable name is 30 characters.
5. Multiple equations may be entered on one line if they are separated by a semi-colon (;)[1]. The maximum line length is 255 characters.
6. The caret symbol ^ or ** is used to indicate raising to a power.
7. The order in which the equations are entered does not matter.
8. The position of knowns and unknowns in the equation does not matter.

After entering the equations for this problem and (optionally) checking the syntax using the Check/Format command in the Calculate menu, the Equations window will appear as shown. Comments are normally displayed in blue on a color monitor. Other formatting options are set with the Preferences command in the Options menu.

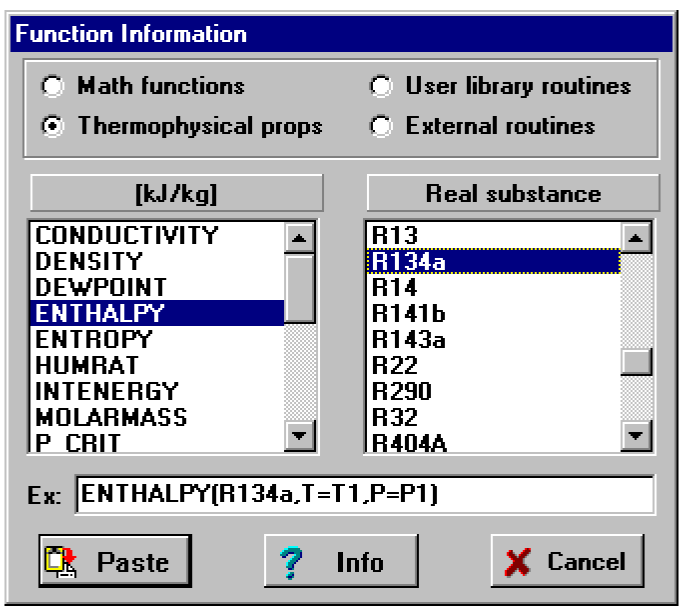


[1] If a comma is selected as the Decimal Symbol in the Windows Regional Settings Control Panel, EES will recognize the comma (rather than a decimal point) as a decimal separator, the semicolon (rather than the comma) as an argument separator, and the vertical bar | (rather than the semicolon) as the equation separator.

Note the use of the **Convert** function in this example to convert the units of the specific kinetic energy [m^2/s^2] to the units used for specific enthalpy [kJ/kg]. The **Convert** function is most useful in these problems. See Chapter 4 for a detailed description of its use.

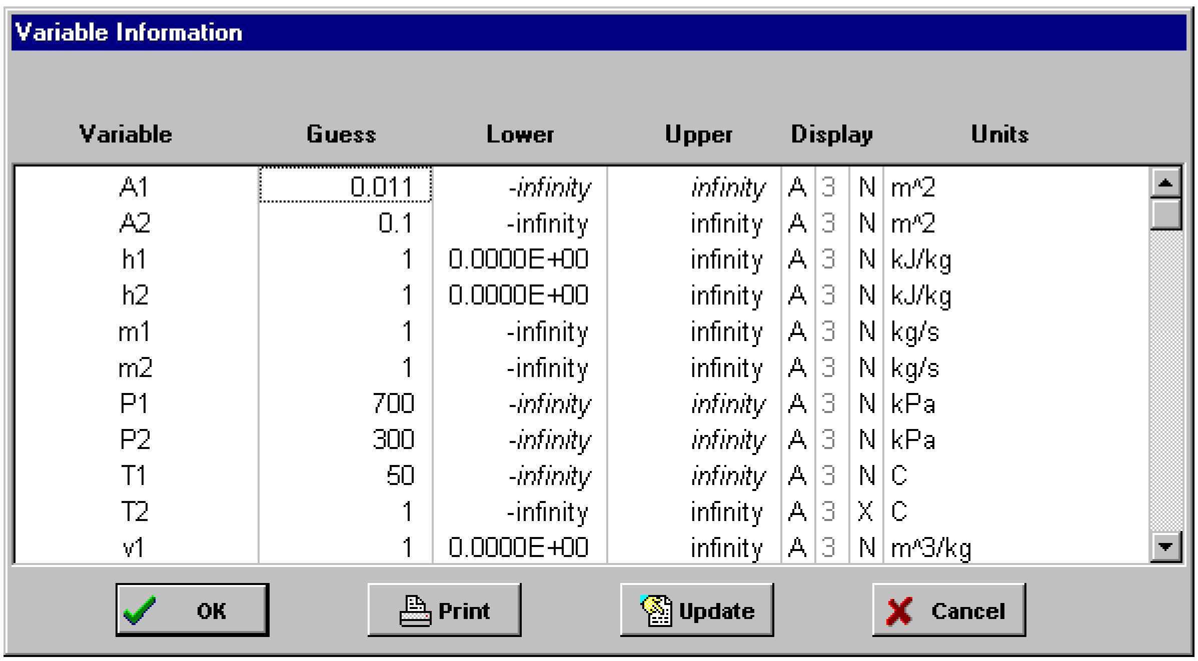
The thermodynamic property functions, such as **enthalpy** and **volume** require a special format. The first argument of the function is the substance name, R134a in this case. The following arguments are the independent variables preceded by a single identifying letter and an equal sign. Allowable letters are T, P, H, U, S, V, and X, corresponding to temperature, pressure, specific enthalpy, specific internal energy, specific entropy, specific volume, and quality. (For psychrometric functions, additional allowable letters are W, R, D, and B, corresponding to humidity ratio, relative humidity, dewpoint temperature, and wetbulb temperature.)

An easy way to enter functions, without needing to recall the format, is to use the Function Information command in the Options menu. This command will bring up the dialog window shown below. Click on the 'Thermophysical props' radio button. The list of built-in thermophysical property function will appear on the left with the list of substances on the right. Select the property function by clicking on its name, using the scroll bar, if necessary, to bring it into view. Select a substance in the same manner. An example of the function showing the format will appear in the Example rectangle at the bottom. The information in the rectangle may be changed, if needed. Clicking the Paste button will copy the Example into the Equations window at the cursor position. Additional information is available by clicking the Info button.



It is usually a good idea to set the guess values and (possibly) the lower and upper bounds for the variables before attempting to solve the equations. This is done with the Variable Information command in the Options menu. Before displaying the Variable Information dialog, EES checks syntax and compiles newly entered and/or changed equations, and then solves all equations with one unknown. The Variable Information dialog will then appear.

The Variable Information dialog contains a line for each variable appearing in the Equations window. By default, each variable has a guess value of 1.0 with lower and upper bounds of negative and positive infinity. (The lower and upper bounds are shown in italics if EES has previously calculated the value of the variable. In this case, the Guess value column displays the calculated value. These italicized values may still be edited, which will force EES to recalculate the value of that variable.)

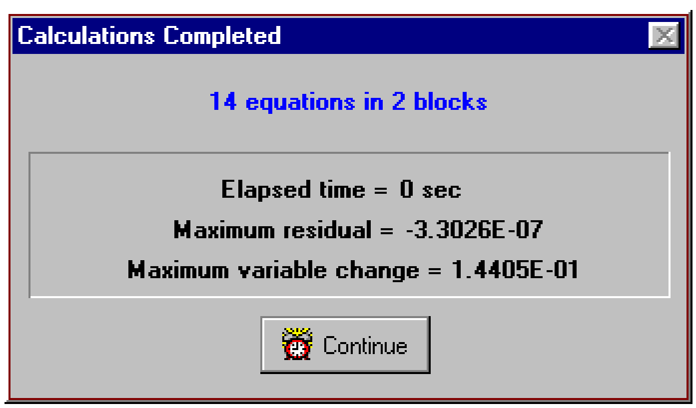


Variable	Guess	Lower	Upper	Display			Units
A1	0.011	*-infinity*	*infinity*	A	3	N	m^2
A2	0.1	-infinity	infinity	A	3	N	m^2
h1	1	0.0000E+00	infinity	A	3	N	kJ/kg
h2	1	0.0000E+00	infinity	A	3	N	kJ/kg
m1	1	-infinity	infinity	A	3	N	kg/s
m2	1	-infinity	infinity	A	3	N	kg/s
P1	700	*-infinity*	*infinity*	A	3	N	kPa
P2	300	*-infinity*	*infinity*	A	3	N	kPa
T1	50	*-infinity*	*infinity*	A	3	N	C
T2	1	-infinity	infinity	A	3	X	C
v1	1	0.0000E+00	infinity	A	3	N	m^3/kg

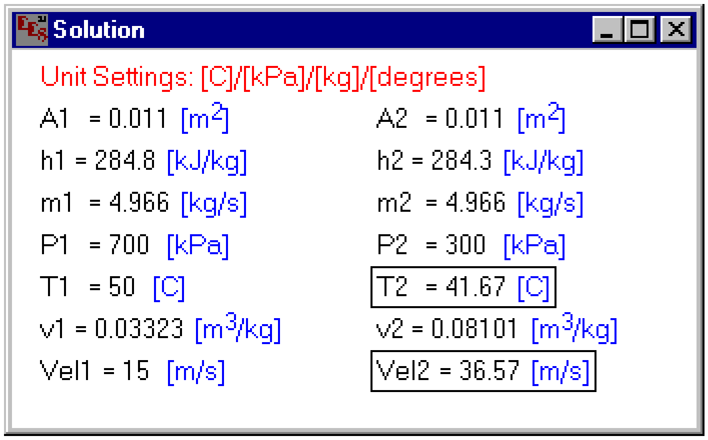
The A in the Display options column indicates that EES will automatically determine the display format for numerical value of the variable when it is displayed in the Solution window. In this case, EES will select an appropriate number of digits, so the digits column to the right of the A is disabled. Automatic formatting is the default. Alternative display options are F (for fixed number of digits to the right of the decimal point) and E (for exponential format). The display and other defaults can easily be changed with the Default Information command in the Options menu, discussed in Chapter 3. The third Display options column controls the hilighting effects such as normal (default), bold, boxed. The units of the variables can be specified, if desired. The units will be displayed with the variable in the Solution window and/or in the Parametric Table. EES does not automatically do unit conversions but it can provide unit conversions using the **Convert** function (Chapter 4). The units information entered here is only for display purposes.

With nonlinear equations, it is sometimes necessary to provide reasonable guess values and bounds in order to determine the desired solution. (It is not necessary for this problem.) The bounds of some variables are known from the physics of the problem. In the example problem, the enthalpy at the outlet, h2, should be reasonably close to the value of h1. Set its guess value to 100 and its lower bound to 0. Set the guess value of the outlet specific volume, v2, to 0.1 and its lower bound to 0. Scroll the variable information list to bring Vel2 into view. The lower bound of Vel2 should also be zero.

To solve the equation set, select the Solve command from the Calculate menu. An information dialog will appear indicating the elapsed time, maximum residual (i.e., the difference between the left-hand side and right-hand side of an equation) and the maximum change in the values of the variables since the last iteration. When the calculations are completed, EES displays the total number of equations in the problem and the number of blocks. A block is a subset of equations which can be solved independently. EES automatically blocks the equation set, whenever possible, to improve the calculation efficiency, as described in Appendix B. When the calculations are completed, the button will change from Abort to Continue.

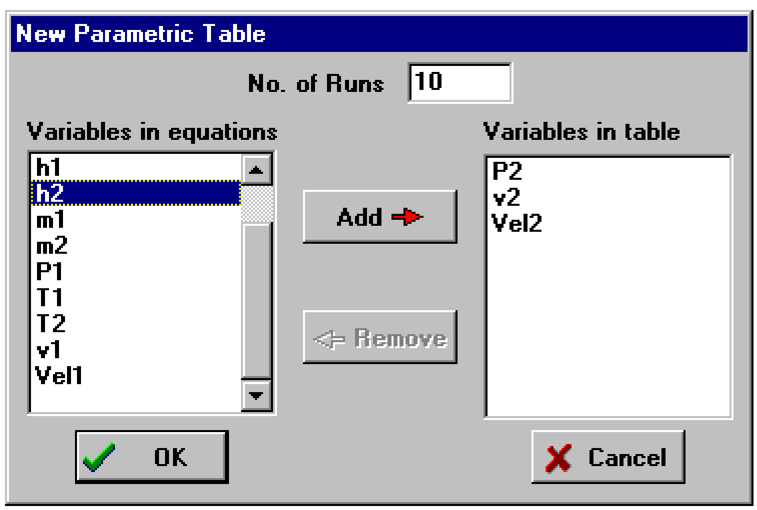


By default, the calculations are stopped when 100 iterations have occurred, the elapsed time exceeds 60 sec, the maximum residual is less than 10^{-6} or the maximum variable change is less than 10^{-9}. These defaults can be changed with the Stop Criteria command in the Options menu. If the maximum residual is larger than the value set for the stopping criteria, the equations were not correctly solved, possibly because the bounds on one or more variables constrained the solution. Clicking the Continue button will remove the information dialog and display the Solution window shown on the next page. The problem is now completed since the values of T2, m2, and Vel2 are determined.



One of the most useful features of EES is its ability to provide parametric studies. For example, in this problem, it may be of interest to see how the throttle outlet temperature and outlet velocity vary with outlet pressure. A series of calculations can be automated and plotted using the commands in the Tables menu.

Select the New Table command. A dialog will be displayed listing the variables appearing in the Equations window. In this case, we will construct a table containing the variables P2, T2, Vel2, and h2. Click on P2 from the variable list on the left. This will cause P2 to be highlighted and the Add button will become active.



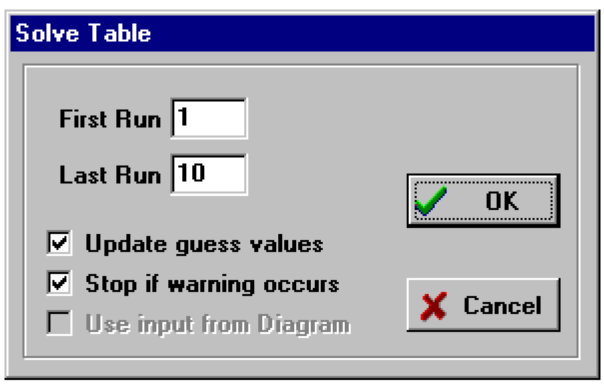
Now click the Add button to move P2 to the list of variables on the right. Repeat for T2, h2, and Vel2, using the scroll bar to bring the variable into view if necessary. (As a short cut, you can double-click on the variable name in the list on the left to move it to the list on the right.). The table setup dialog should now appear as shown above. Click the OK button to create the table.

The Parametric Table works much like a spreadsheet. You can type numbers directly into the cells. Numbers which you enter are shown in black and produce the same effect as if you set the variable to that value with an equation in the Equations window. Delete the P2 = 300 equation currently in the Equations window or enclose it in comment brackets { }. This equation will not be needed because the value of P2 will be set in the table. Now enter the values of P2 for which T2 is be determined. Values of 100 to 550 have been chosen for this example. (The values could also be automatically entered using Alter Values in the Tables menu or by using the Alter Values control ▼ at the upper right of each table column header, as explained in Chapter 2.) The Parametric Table should now appear as shown below.

Parametric Table

	1 P2 [kPa]	2 T2 [C]	3 Vel2 [m/s]	4 h2 [kJ/kg]
Run 1	100			
Run 2	150			
Run 3	200			
Run 4	250			
Run 5	300			
Run 6	350			
Run 7	400			
Run 8	450			
Run 9	500			
Run 10	550			

Now, select Solve Table from the Calculate menu. The Solve Table dialog window will appear allowing you to choose the runs for which the calculations will be done.

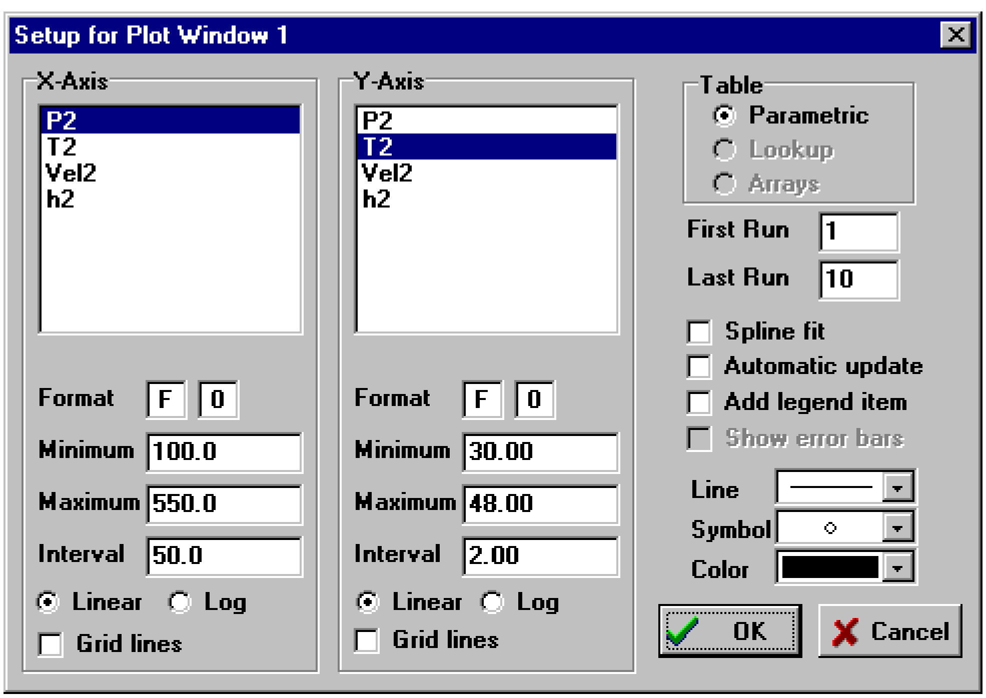


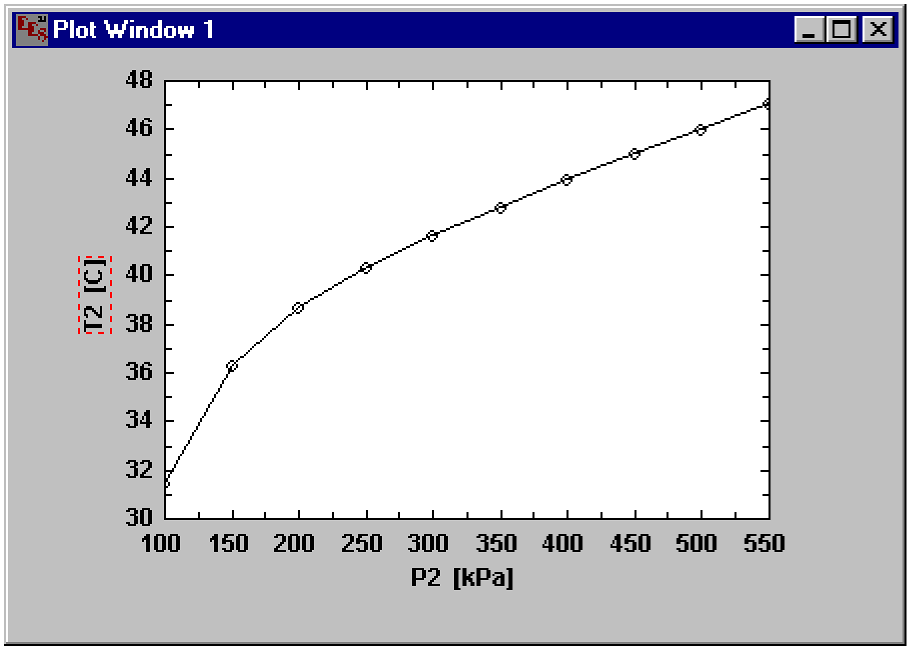
When the Update Guess Values control is selected, as shown, the solution for the last run will provide guess values for the following run. Click the OK button. A status window will be displayed, indicating the progress of the solution. When the calculations are completed, the values of T2, Vel2, and h2 will be entered into the table. The values calculated by EES will be displayed in blue, bold or italic type depending on the setting made in the Screen Display tab of the Preferences dialog window in the Options menu.

The relationship between variables such as P2 and T2 is now apparent, but it can more clearly be seen with a plot. Select New Plot Window from the Plot menu. The New Plot Window dialog window shown below will appear. Choose P2 to be the x-axis by clicking on P2 in the x-axis list. Click on T2 in the y-axis list. Select the scale limits for P2 and T2, and set the number of divisions for the scale as shown. Grid lines make the plot easier to read. Click on the Grid Lines control for both the x and y axes. When you click the OK button, the plot will be constructed and the plot window will appear as shown.

Parametric Table

	1 P2 [kPa]	2 T2 [C]	3 Vel2 [m/s]	4 h2 [kJ/kg]
Run 1	100	31.46	109.9	278.9
Run 2	150	36.32	73.79	282.2
Run 3	200	38.7	55.29	283.4
Run 4	250	40.34	44.08	284
Run 5	300	41.67	36.57	284.3
Run 6	350	42.86	31.19	284.4
Run 7	400	43.97	27.15	284.6
Run 8	450	45.03	24	284.6
Run 9	500	46.06	21.48	284.7
Run 10	550	47.07	19.42	284.7





Once created, there are a variety of ways in which the appearance of the plot can be changed as described in the Plot Windows section of Chapter 2 and in the Plot menu section of Chapter 3.

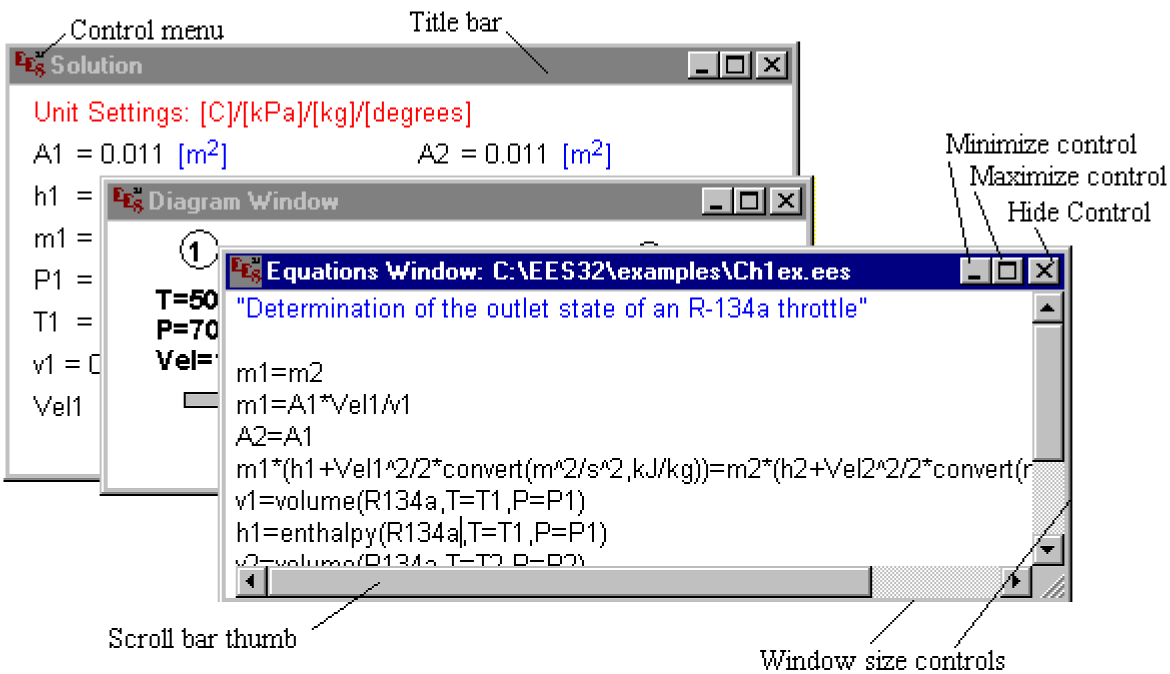
This example problem illustrates some of the capabilities of EES. With this example behind you, you should be able to solve many types of problems. However, EES has many more capabilities and features, such as curve-fitting, uncertainty analyses, complex variables, arrays.

CHAPTER 2

EES Windows

General Information

The information concerning a problem is presented in a series of windows. Equations and comments are entered in the Equations window. After the equations are solved, the values of the variables are presented in the Solution and Arrays windows. The residuals of the equations and the calculation order may be viewed in the Residuals window. Additional windows are provided for the Parametric and Lookup Tables, a diagram and up to 10 plots. There is also a Debug window. A detailed explanation of the capabilities and information for each window type is provided in this section. All of the windows can be open (i.e., visible) at once. The window in front is the active window and it is identified by its highlighted (black) title bar. The figure below shows the appearance of the EES windows in Microsoft Windows 95 and NT 4.0. The appearance may be slightly different in other Windows versions.



One difference between EES and most other applications is worth mentioning. The Close control merely hides a window; it does not delete it. Once closed, a window can be reopened (i.e., made visible) by selecting it from the Windows menu.

Every window has a number of controls.

1. To move the window to a different location on the screen, move the cursor to a position on the title bar of the window and then press and hold the left button down while sliding the mouse to a new location.

2. To hide the window, select the Close command (or press Ctrl-F4) from the control menu box at the upper left of the window title bar. (Windows 95 and NT 4.0 also provide a Close icon at the upper right of the title bar.) You can restore a hidden window by selecting it from the Windows menu.

3. The Maximize box at the upper right of the window title bar causes the window to be resized so as to fill the entire screen. The Restore box with an up and down arrow will appear below the Maximize box. Click the Restore box (or select Restore form the Control menu box) to return the window to its former size.

4. The size of any window can be adjusted using the window size controls at any border of the window. To change the size of any window, move the cursor to the window border. The cursor will change to a horizontal or vertical double arrow. Then press and hold the left button down while moving the mouse to make the window larger or smaller. Scroll bars will be provided if the window is made too small to accommodate all the information.

5. Double-clicking the left mouse button on the EES icon at the upper left of the title bar will hide that window.

6. Use the Cascade command in the Windows menu to move and resize all open windows.

Equations Window

The Equations window operates very much like a word processor. The equations which EES is to solve are entered in this window. Editing commands, i.e., Cut, Copy, Paste, are located in the Edit menu and can be applied in the usual manner. Additional information relevant to the Equations window follows.

1. Blank lines may be used to make the Equations window more legible. Comments are enclosed in braces {comment} or in quote marks "another comment" and may span multiple lines. Nested comment fields within braces are permitted. Comments within quote marks will appear in the Formatted Equations window.

2. Equations may be entered in any order. The order of the equations has no effect on the solution, since EES will block the equations and reorder them for efficient solution as described in Appendix B.

3. The order of mathematical operators used in the equations conform to the rules used in FORTRAN, Basic, C or Pascal. For example, the equation

   X = 3 + 4 * 5

   will result in X having a value of 23. The caret symbol ^ or ** can be used to indicate raising to a power. Arguments of functions are enclosed in parentheses. EES does not require a variable to appear by itself on the left-hand side of the equation, as does FORTRAN and most other programming languages. The above equation could have been entered as

   (X – 3) / 4 = 5

4. Upper and lower case letters are not distinguished. EES will (optionally) change the case of all variables to match the manner in which they first appear in the Equations window depending on the settings selected in Preferences dialog in the Options menu. However, this change is made only when an equation is first compiled or modified or when Check/Format command in the Calculate menu is issued.

5. Variable names must start with a letter and consist of any keyboard characters except (‘|)*/+-^{ } ":;. The maximum variable length is 30 characters. String variables hold character information and are identified with a $ as the last character in their names, as in BASIC. Array variables are identified with square braces around the array index or indices, e.g., X[5,3]. The quantity within the braces must be a number, except within the scope of the sum, product or Duplicate commands. As a general rule, variables should not be given names which correspond to those of built-in functions (e.g., **pi**, **sin**, **enthalpy**).

6. EES has an upper limit of 5000 variables (32-bit version).

7. Equations are normally entered one per line, terminated by pressing the Return or Enter keys. Multiple equations may be entered on one line if they are separated by a semi-colon[2]. Long equations are accommodated by the provision of a horizontal scroll bar which appears if any of the equations is wider than the window. However, each equation must be less than 255 characters.

8. EES compiles equations into a compact stack-based form. The compiled form is saved in memory so that an equation needs to be compiled only when it is first used or when it is changed. Any error detected during the compilation or solution process will result in an explanatory error message and highlighting of the line in which the problem was discovered.

9. Equations can be imported or exported from/to other applications by using Cut, Copy and Paste commands in the Edit menu. The Merge and Load Library commands in the File menu and the $INCLUDE directive may also be used to import the equations from an existing file. The Merge command will import the equations from an EES or text file and place them in the Equations window at the cursor position. Equations imported with the $INCLUDE directive will not appear in the Equations window.

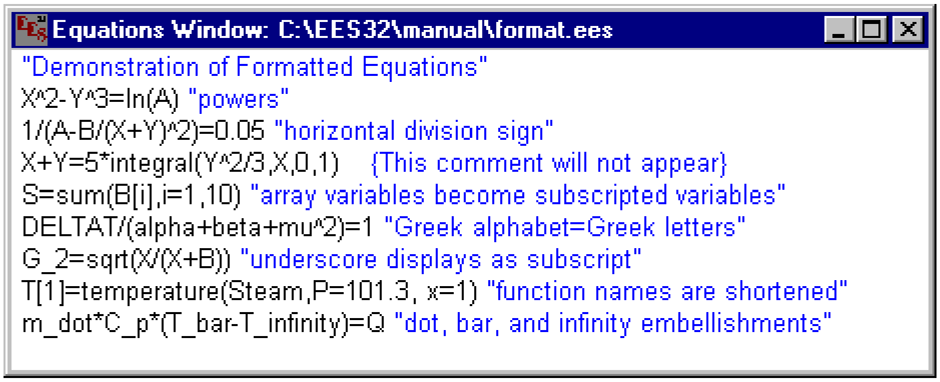
10. Clicking the right mouse button in the Equations window will either insert or remove curly brace comments around the selected text. If the selected text is already commented, i.e., begins with a left brace and ends with a right brace, the comments will be removed - otherwise the braces will be inserted.

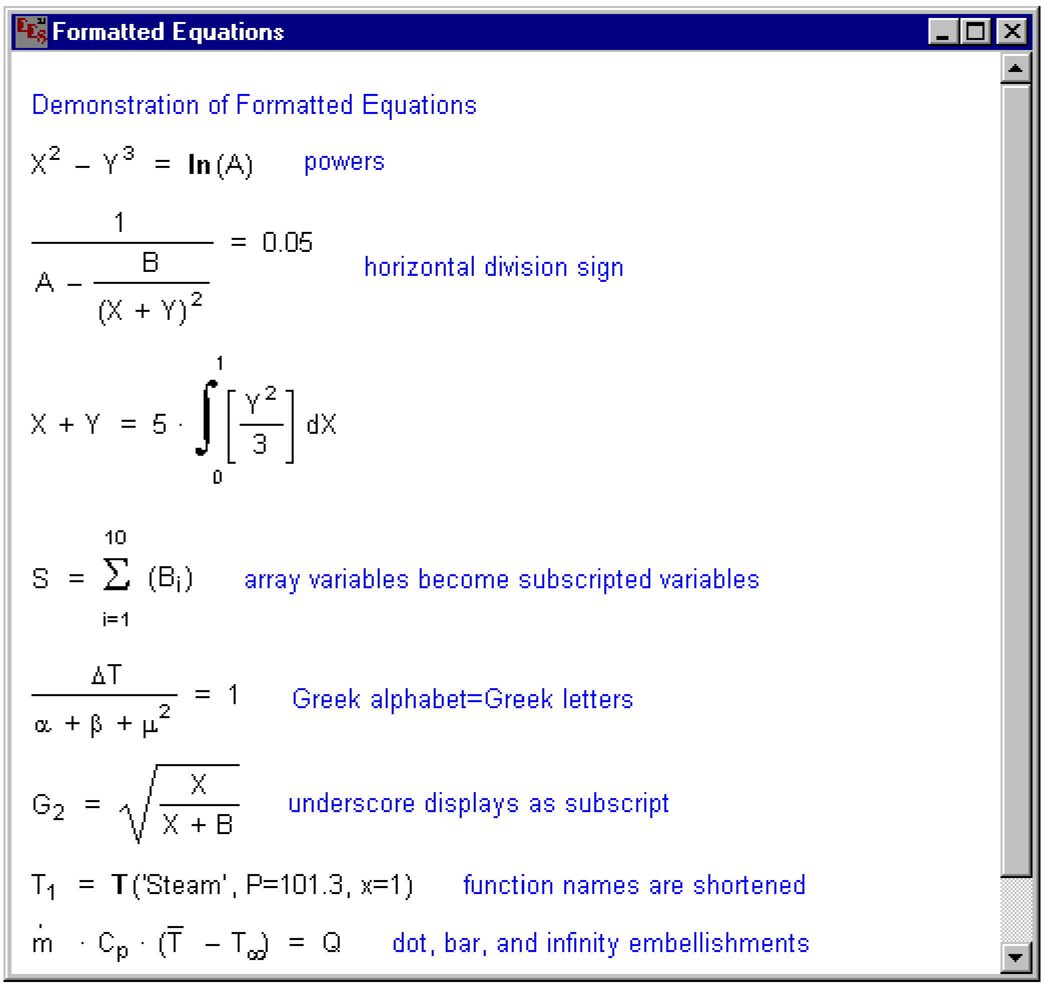
11. If EES is configured to operate in complex mode, all variables as assumed to have real and imaginary components. The complex mode configuration can be changed in the Preferences Dialog (Options menu) or with the $Complex On/Off directive.

[2] If a comma is selected as the Decimal Symbol in the Windows Regional Settings Control Panel, EES will recognize the comma (rather than a decimal point) as a decimal separator, the semicolon (rather than the comma) as an argument separator, and the vertical bar | (rather than the semicolon) as the equation separator.

Formatted Equations Window

The Formatted Equations window displays the equations entered in the Equations window in an easy-to-read mathematical format as shown in the sample windows below.





Note that comments appearing in quotes in the Equations window are displayed in the Formatted Equations window but comments in braces are not displayed. An examination of the Formatted Equations Window will reveal a number of EES features to improve the display, in addition to the mathematical notation. Array variables, such as B[1] are (optionally) displayed as subscripted variables. Sums and integrals are represented by their mathematical signs. If a variable name contains an underscore, the underscore will signify the beginning of a subscript, as in variable G_2. However, note that although G[2] and G_2 will display in the same manner in the Formatted Equations Window, they are different variables with different properties. The index of array variables, e.g., G[2], can be used within the scope of Duplicate statements, or with the Sum and Product functions. In addition, the calculated value of G[2] can be displayed in the Arrays Window, as described in more detail in this chapter.

Placing _dot or _bar after a variable name places a dot or bar centered over the name. The _infinity results in a subscript with the infinity symbol (∞). Variables having a name from the Greek alphabet are displayed with the equivalent Greek letter. For example, the variable name beta will display as ß and mu will display as a µ. If the variable name in the Equations window is entered <u>entirely</u> in capital letters, and if the capital Greek letter is distinct from the English alphabet, the capital Greek letter will be used. For example, the variable name GAMMA will be displayed as Γ. A special form is provided for variables beginning with DELTA. For example, DELTAT displays as ΔT. Capital BETA looks just like a B, so EES will display the lower case equivalent, i.e., ß.

The formatted equations and comments appearing the Formatted Equations window can be moved to other positions if you wish. To move an equation or comment, move the cursor to the item and then press and hold the left mouse button down while sliding the equation or comment to a new location.

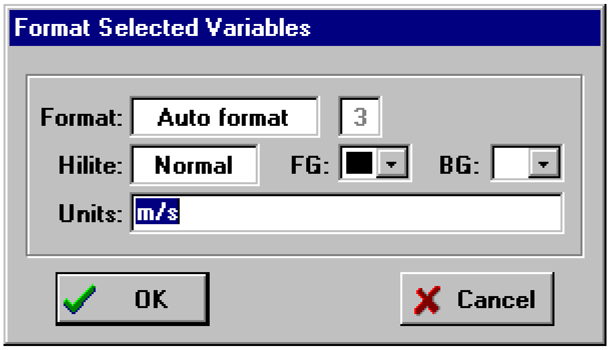
The formatted equations and comments are internally represented as Windows MetaFilePict items or pictures. You can copy one or more equation pictures from this window to other applications (such as a word processor or drawing program). To copy an equation, first select it by clicking the left mouse button anywhere within the equation rectangle. A selected equation or comment will be displayed in inverse video. You may select additional equations. Alternatively, the Select Display command in the Edit menu can be used to select all of the equations and comments which are currently visible in the Formatted Equations window. Copy the selected equations and comments to the clipboard with the Copy command. The equations will be unselected after the copy operation. Comments normally appear in blue text on the Formatted Equations window and they will appear in color when copied to the Clipboard. If you wish to have the comments displayed in black, hold the Shift key down while issuing the Copy command.

The text in the Formatted Equations window can not be edited. However, clicking the right mouse button on an equation in the Formatted Equation window will bring the Equations window to the front with that equation selected where it can be edited.

Solution Window

The Solution window will automatically appear in front of all other windows after the calculations, initiated with the Solve or Min/Max commands in the Calculate menu, are completed. The values and units of all variables appearing in the Equations window will be shown in alphabetical order using as many columns as can be fit across the window.

The format of the variables and their units can be changed using the Variable Info command in the Options menu, or more simply, directly from the Solution window. Clicking the left mouse button on a variable selects that variable which is then displayed in inverse video. Clicking the left mouse button on a selected variable unselects it. Double-clicking the left mouse button (or clicking the right mouse button) brings up the Format Variable dialog window. The changes made in the Format Variable dialog are applied to ALL selected variables. Pressing the Enter key will also bring up the Format Variable dialog window.



The numerical format (style and digits) and the units of the selected variables can be selected in this dialog window. When configured in Complex mode, an additional formatting option is provided for displaying the variable in rectangular or polar coordinates. The selected variables can also be highlighted (with underlining, bold font, foreground (FG) and background (BG) colors, etc.) or hidden from the Solution window. If a variable is hidden, it can be made visible again with the Display controls in the Variable Info dialog window. Additional information pertaining to the operation of the Solution window follows.

1. The Solution window is accessible only after the calculations are completed. The Solution menu item in the Windows menu will be dimmed when the Solution window is not accessible.

2. The unit settings made with the Unit System command in the Options menu will be displayed at the top of the Solution window if any of the built-in thermophysical property or trigonometric functions are used.

3. The Solution window will normally be cleared and hidden if any change is made in the Equations window. However, there is an option in the Preferences dialog of the Options menu to allow the Solution window to remain visible.

4. The number of columns displayed on the screen can be altered by making the window larger or smaller.

5. If EES is unable to solve the equation set and terminates with an error, the name of the Solution window will be changed to Last Iteration Values and the values of the variables at the last iteration will be displayed in the Solution window.

6. When the Solution window is foremost, the Copy command in the Edit menu will appear as Copy Solution. The Copy Solution command will copy the selected variables (shown in inverse video) to the clipboard both as text and as a picture. The text will provide for each variable (selected or not) a line containing the variable name, its value, and its units. The picture will show only those variables which are selected in the same format as they appear in the Solution window. The Select Display command in the Edit menu will select all variables currently visible in the Solution Window. (If you wish to force a black and white picture, hold the Shift key down when you issue the Copy Solution command.) Both the text or the picture can be pasted into another application, such as a word processor. Most word processors will, by default, paste the text. To paste the picture instead of the text, select the Paste Special command and select picture.

7. If the ☒ Display subscripts and Greek symbols option in the General Display tab of the Preferences dialog is selected, EES will display subscripts and superscripts of variable units. For example, m^2 will appear as m^2. An underscore character is used to indicate a subscript so lb_m will appear as lb_m.

Arrays Window

EES allows the use of array variables. EES array variables have the array index in square brackets, e.g., X[5] and Y[6,2]. In most ways, array variables are just like ordinary variables. Each array variable has its own guess value, lower and upper bounds and display format. However, simple arithmetic operations are supported for array indices so array variables can be more convenient in some problems as discussed in Chapter 7.

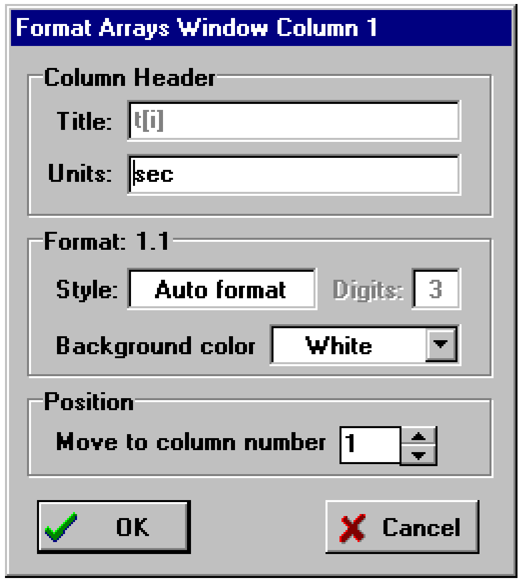
The values of all variables including array variables are normally displayed in the Solution window after calculations are completed. However, array variables may optionally be displayed in a separate Arrays window, rather than in the Solution window. This option is controlled with the ☒ Place array variables in the Arrays window check box in the Preferences dialog (Options tab) in the Options menu. If this option is selected, an Arrays window such as that shown below will automatically be produced after calculations are completed showing all array values used in the problem in alphabetical order with the array index value in the first column.

Arrays Table

	1 t_i [sec]	2 y_i [observed]	3 $y_{P,i}$ [predicted]
[1]	1.1	3	2.998
[2]	1.2	2.9	3.16
[3]	1.3	3.6	3.322
[4]	1.9	4.2	4.297
[5]	2.3	5.1	4.948
[6]	3.1	5.9	6.257
[7]	3.3	7	6.586
[8]	4.1	7.8	7.903
[9]	4.4	8	8.399
[10]	4.6	9.1	8.73

The values in the Arrays window may be plotted using the New Plot Window command in the Plot menu. Part or all of the data in the Arrays window can be copied to another application by selecting the range of cells of interest followed by use of the Copy command in the Edit menu. If you wish to include the column name and units along with the numerical information in each column, hold the Ctrl key down while issuing the Copy command.

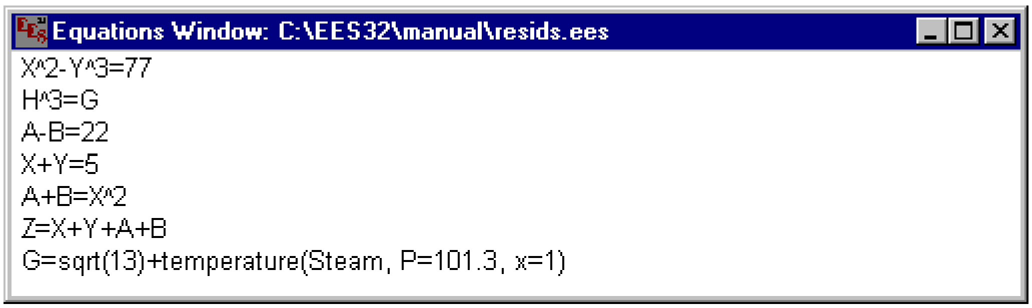
The format of values in any column of the Arrays window can be changed by clicking the left mouse button on the array name at the top of the column. The following dialog window will appear in which the units, display format and column position can be changed. Note that you can enter a number in the column number field or use the up/down arrows to change its value. If the value you enter is greater than the number of columns in the table, the column will be positioned at the right of the table.



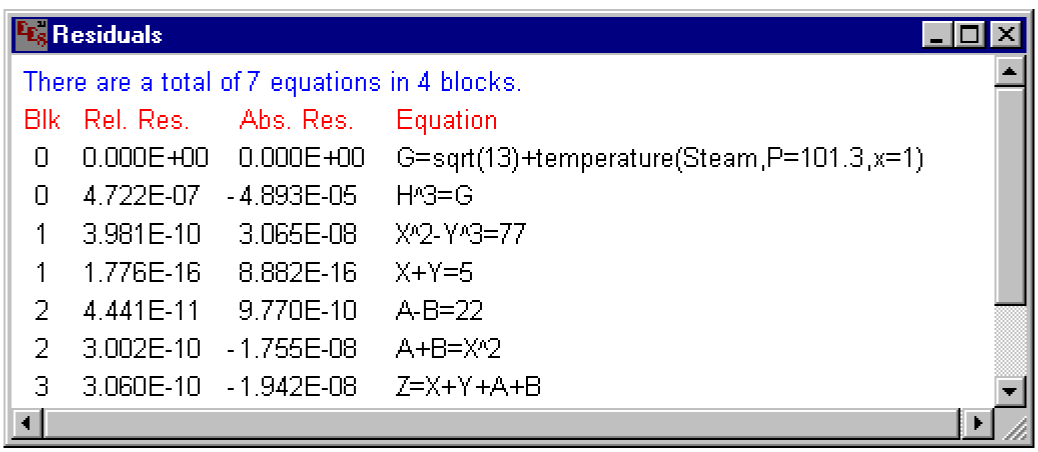
Residuals Window

The Residuals window indicates the equation blocking and calculation order used by EES, in addition to the relative and absolute residual values. The absolute residual of an equation is the difference between the values on the left and right hand sides of the equation. The relative residual is the magnitude of the absolute residual divided by the value of left side of the equation.[3] The relative residuals are monitored during iterative calculations to determine when the equations have been solved to the accuracy specified with the Stopping Criteria command in the Options menu.

Consider, for example, the following set of six equations and six unknowns.



EES will recognize that these equations can be blocked, i.e., broken into two or more sets, as described in more detail in Appendix B. The blocking information is displayed in the Residuals window.



Variables having values which can be determined directly, i.e., without simultaneously finding the values of other variables, such as G in the example above, are determined first

[3] If the value of the left hand side of an equation is zero, the absolute and relative residuals assume the same value.

and assigned to Block 0[4]. Once G is known, H can be determined. The order in which these individual equations are solved in Block 0 is indicated by the order in which they appear in the Residuals window. After solving all equations in Block 0, EES will simultaneously solve the equations in Block 1, then Block 2, and so on until all equations are solved. The first and third equations in the example above can be solved independently of other equations to determine X and Y and are thereby placed in Block 1. Similarly, the second and fourth equations which determine A and B are placed in Block 2. With X, Y, A, and B now known, Z can be determined, so it appears in Block 3.

The Residuals window will normally be hidden when any change is made in the Equations window. This automatic hiding can be disabled with the Display Options command in the Options menu.

It is possible to display the Residuals window in a debugging situation. If the number of equations is less than the number of unknowns, EES will not be able to solve the equation set, but the Residuals window can be made visible by selecting it from the Windows menu. Normally, the block numbers appear in sequential order. When one or more equations are missing, EES will skip a block number at the point in which it encounters this problem. The equations in the following blocks should be carefully reviewed to determine whether they are correctly and completely entered.

The information in the Residuals window is useful in coaxing a stubborn set of equations to converge. An examination of the residuals will indicate which equations have been solved by EES and which have not. In this way, the block of equations which EES could not solve can be identified. Check these equations to be sure that there is a solution. You may need to change the guess values or bounds for the variables in this block using the Variable Info command in the Options menu.

Doubling-clicking the left mouse button (or clicking the right mouse button) on an equation in the Residuals window will cause the Equations window to be brought to the front with the selected equation highlighted.

The entire contents of the Residuals window will be copied as tab-delimited text to the Clipboard if the Copy command is issued when the Residuals window is foremost.

---

[4] Variables specified in the Diagram window are identified with a D rather than a block number. See the Diagram Window section. In Complex mode, each equation is shown twice, once for the real part identified with (r) and again for the imaginary component labeled with (i)

Parametric Table Window

Parametric Table

	1 P2 [kPa]	2 T2 [C]	3 Vel2 [m/s]	4 h2 [kJ/kg]
Run 1	100	31.46	109.9	278.9
Run 2	150	36.32	73.79	282.2
Run 3	200	38.7	55.29	283.4
Run 4	250	40.34	44.08	284
Run 5	300	41.67	36.57	284.3

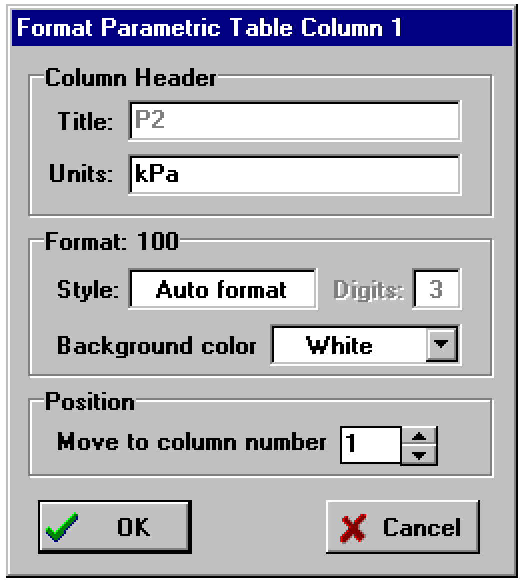
The Parametric Table window contains the Parametric Table which operates somewhat like a spreadsheet. Numerical values can be entered into any of the cells. Entered values, e.g., the values in the P2 column in the above table, are assumed to be independent variables and are shown in normal type in the font and font size selected with the Preferences command (Options menu). Entering a value in the Parametric Table produces the same effect as setting that variable to the value with an equation in the Equations window. Dependent variables will be determined and displayed in the table in blue, bold type, or italics (depending on the choice made with the Preferences command) when the Solve Table or Min/Max Table command in the Calculate menu is issued.

1. A table is generated using the New Parametric Table command in the Tables menu. The variables which are to appear in the table are selected from a list of variables currently appearing in the Equations window.

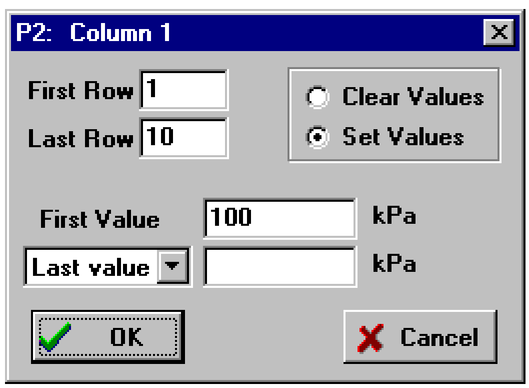
2. Each row of the Parametric Table is a separate calculation. The number of rows is selected when the table is generated, but may be altered using the Insert/Delete Runs command in the Tables menu.

3. Variables may be added to or deleted from an existing Parametric Table using the Insert/Delete Vars command in the Tables menu.

4. The initial order in which the columns in the Parametric Table appear is determined by the order in which the variables in the table were selected in the New Parametric Table dialog. To change the column number order, click the left mouse button in the column header cell (but not on the alter values control at the upper right). A dialog window will appear as shown below in which the column number can be changed be clicking the up or down arrows to the right of the column number or by directly editing the column number. The display format, units, and column background color can also be entered or changed at this point.



5. Values can be automatically entered into the Parametric table using the Alter Values command in the Tables menu. Alternatively, clicking the mouse on the ▼ control at the upper right of the column header cell will bring up the dialog window shown below which provides the same automatic entry somewhat more conveniently.



6. A Sum row which displays the sum of the values in each column may be hidden or made visible using the 'Include a Sum row in the Parametric table' control provided in the Preferences dialog window (Options tab) in the Options menu.

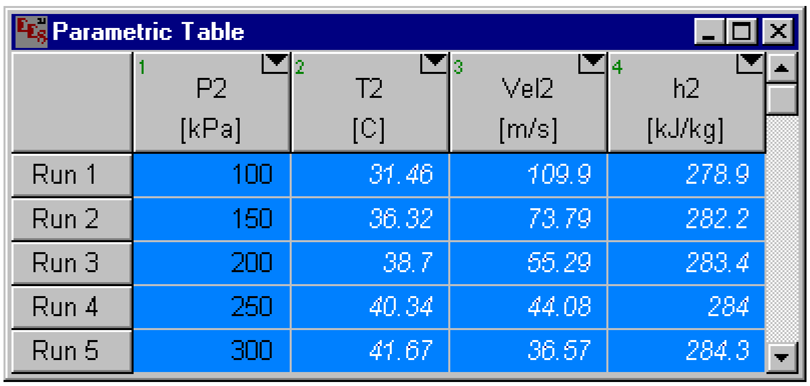
7. A Parametric Table is used to solve differential equations or integrals. See Chapter 7 for additional information.

8. The **TableValue** function returns the value of a table cell at a specified row and column.

9. The TableRun# function returns the row of the table for which calculations are currently in progress.

10. The independent variables in the Parametric Table may differ from one row to the next. However, when the independent variables are the same in all rows, EES will not have to recalculate the Jacobian and blocking factor information and can thus do the calculations more rapidly.

11. Tabular data may be imported or exported from the Parametric Table via the Clipboard using the Copy and Paste commands in the Edit menu. To copy data from any of the EES tables, click the mouse in the upper left cell. Hold the Shift key down and click in the lower right cell, using the scroll bar as needed. The selected cells will be shown in inverse video. When the Shift key is released, the upper left cell which has the focus will return to normal display. However, even though it is not displayed in inverse video, the upper left cell is selected and it will be placed on the clipboard with other cells when the Copy command is issued. Use the Select All command in the Edit menu to select all of the cells in the table. The data are placed on the clipboard with a tab between each number and a carriage return at the end of each row. With this format, the table data will paste directly into a spreadsheet application. If you wish to include the column name and units along with the numerical information in each column, hold the Ctrl key down while issuing the Copy command



	1 P2 [kPa]	2 T2 [C]	3 Vel2 [m/s]	4 h2 [kJ/kg]
Run 1	100	31.46	109.9	278.9
Run 2	150	36.32	73.79	282.2
Run 3	200	38.7	55.29	283.4
Run 4	250	40.34	44.08	284
Run 5	300	41.67	36.57	284.3

Lookup Table Window

The Lookup Table provides a means of using tabular information in the solution of the equations. A Lookup Table is created using the New Lookup Table command in the Tables menu. The number of rows and columns in the table are specified when the table is created and may be altered with Insert/Delete Rows and Insert/Delete Cols commands in the Tables menu. A Lookup Table may be saved on a disk (separately from the EES file) using the Save Lookup command in the Tables menu. A .LKT filename extension is used to designate EES Lookup files. Lookup files can also be saved in ASCII format with a .TXT filename extension. The Lookup table may then be accessed from other EES programs in either format.

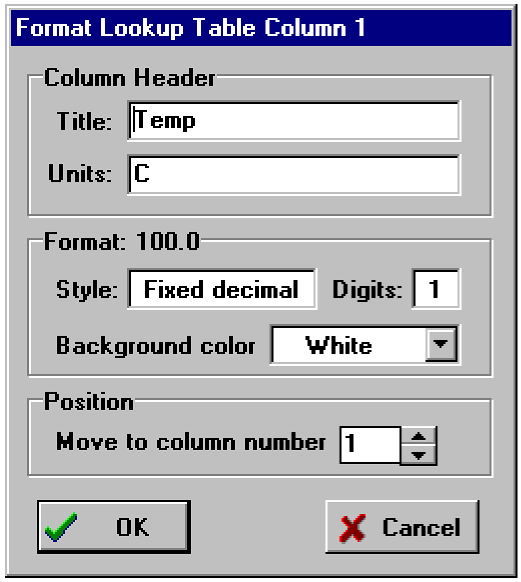
The **Interpolate** commands provide linear, quadratic or cubic interpolation or extrapolation of the data in the Lookup Table. See Chapter 4 for details. In addition, the **Lookup**, **LookupCol**, and **LookupRow** functions allow data in a Lookup Table to be linearly interpolated (forwards and backwards) and used in the solution of the equations. The Lookup Table may either reside in the Lookup Table Window or in a previously-saved Lookup File with a .LKT filename extension, as explained in more detail in Chapter 4.

Lookup Table

	1 Temp [C]	2 Time [sec]	3 Pos [m]
Row 1	100.0	0.00	5.50
Row 2	120.0	1.00	5.86
Row 3	140.0	2.00	6.11
Row 4	160.0	3.00	6.36
Row 5	180.0	4.00	6.58

A sample Lookup Table is shown above. The column number is displayed in small type at the upper left of each column header cell. The column number is needed for use with the Lookup functions. However, the **Lookup** functions will also accept ‘ColumnName’ in place of the column number where ‘ColumnName’ is the name of the column shown in the column heading surrounded by single quote marks.[5] The column names are initially **Column1**, **Column2**, etc. but these default names can be changed by clicking the left mouse button in the header cell which will bring up the following dialog window.

[5] EES will also accept #ColumnName in place of the column number.



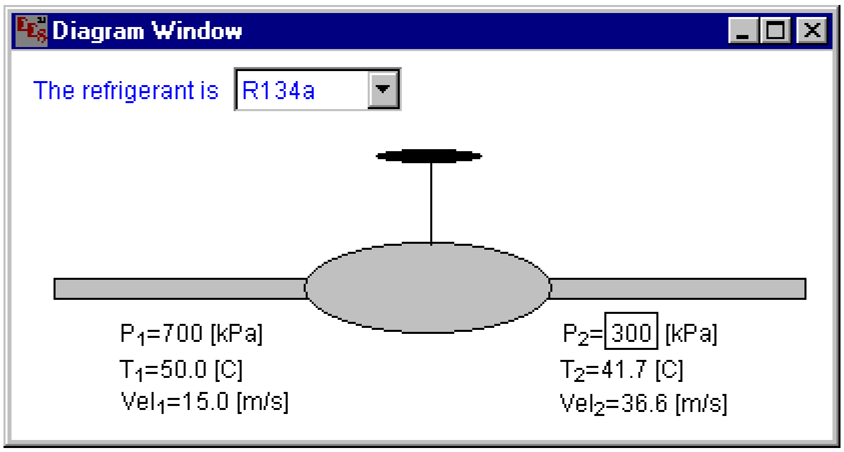
The column title can be changed and units for the values in the column may be specified. The Format controls allow the data in each column of the table to be displayed in an appropriate numerical format. A control is provided to change the background color for each column. The column position may also be changed by either clicking the up or down arrows to the right of the column number or by editing the number directly.

Data can be imported to or exported from the Lookup Table through the Clipboard in the same manner as described for the Parametric Table. Data may be automatically entered into the Lookup table by clicking on the control at the upper right of the column header cell, as described for the Parametric table. Hold the Ctrl key down when issuing the copy command if you wish to copy the column names and units with the other numerical information on the Clipboard. Data may be interchanged between the Parametric and Lookup Table windows. In particular, columns of data in the Parametric Table may be stored in the Lookup Table so that they may be plotted or reused at a later time.

A memory-based Lookup Table can be deleted, if desired, with the Delete Lookup menu item in the Options menu. Lookup Table files saved with a .LKT or .TXT filename extension can not be deleted from within EES.

Diagram Window

The Diagram window can be used in two ways. First, it provides a place to display a diagram (or text) relating to the problem which is being solved. For example, a schematic diagram of a system identifying state point locations can be displayed in the Diagram window to help interpret the equations in the Equations window. Second, the Diagram window can be used for both input and output of information and for report generation. Shown below is a diagram with a schematic and input and output variable information superimposed.

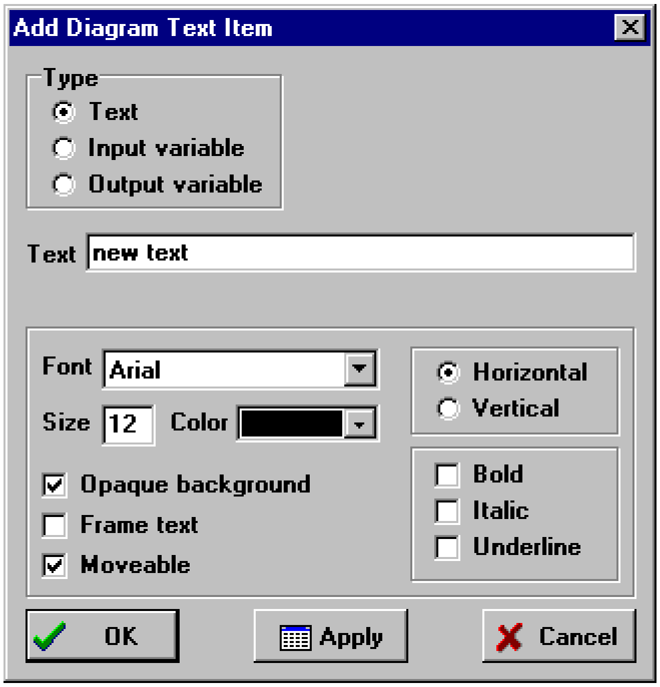


The diagram itself is not drawn in EES, but rather in any drawing program which produces an object drawing such as MSPaint, Microsoft Draw (included in Word for Windows), Corel Draw, Designer, or Power Point. A scanned image may also be placed in the Diagram window. Copy the drawing and then paste it into the Diagram window. The diagram will be saved along with all other problem information.

The diagram can be repositioned in the Diagram window by pressing and holding the left mouse button down anywhere within the diagram rectangle while sliding the diagram to its new location. Any text placed on the Diagram window will move with the diagram itself.

The diagram and all associated text will be scaled to fit within the Diagram Window by double-clicking the left mouse button (or clicking the right mouse button) anywhere within the Diagram window, except on a text item. The diagram can be made larger or smaller by first changing the size of the Diagram window and then double-clicking to change the size of the diagram itself. The aspect ratio of the diagram is not changed as the diagram is resized.

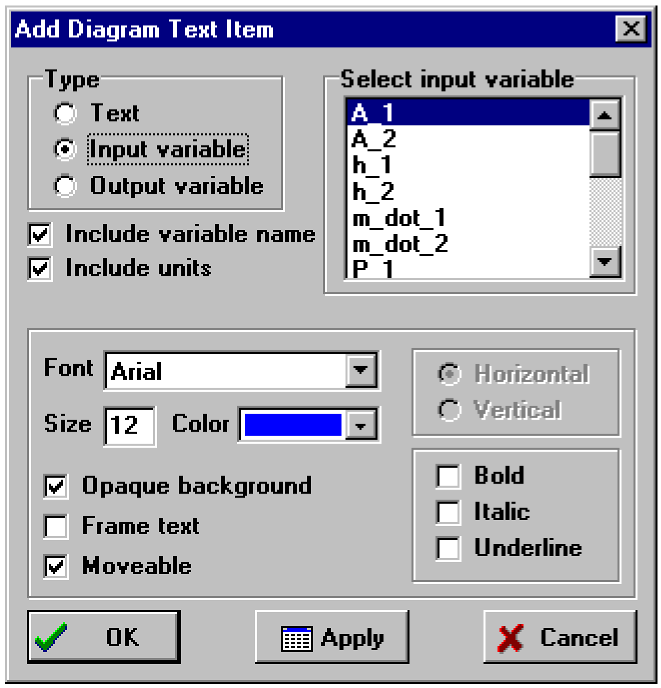
The Add Diagram Text command in the Options menu allows text to be placed anywhere on the Diagram window. Three different types of text may be selected by the radio buttons at the upper left of the diagram window. Selecting the Text radio button will cause the window to appear as shown below, in which the text and its characteristics can be specified. The text will initially appear in a default position within the Diagram window when the dialog is dismissed. It can then be dragged to a new position by pressing and holding the left mouse button down while sliding the text to the desired location. The text or any of its characteristics can later be changed by double-clicking the left mouse button (or by clicking the right mouse button) while the cursor is positioned over the text.



Clicking the Input or Output radio buttons changes the dialog window display so that a list of currently defined variables replaces the text edit box, as shown. Select the variable by clicking on its name in the list. Both Input and Output variables values are displayed on the diagram with the option of also displaying the variable name and unit string. An Output variable displays the value of the selected variable calculated during the previous calculation. An Input variable will be displayed with the value enclosed in a rectangle. This value can be edited and it provides the same function as an equation in the Equations window which sets the variable to a value.

If a string variable (identified with a $ character as the last character in the variable name) is selected for an Input variable, EES will provide an option of selecting the variable from a pull-down list of string constants that you provide. For example, the Diagram window shown at the start of this section employs a pull-down list of refrigerants. The refrigerant selected by the user is assigned to a string variable which is then provided as the fluid name in the thermodynamic functions.

When either the Solve or Min/Max commands (Calculate menu) are issued, EES will first examine the Diagram window to see which variables, if any, are set, provided that the Diagram window is not hidden. A value which is set in the Diagram window cannot also be set in the Equations window. After the calculations are completed, the newly-calculated values of the Output variables will be displayed on the Diagram window. Output values will display as **** if the value is not currently defined.



The Diagram window input is ignored if the Diagram window is hidden. The Diagram window can be used with the Parametric table (i.e., the Solve Table command) if the 'Use Input from Diagram' check box is checked in the Solve Table dialog window.

Use the Clear command in the Edit menu to delete an existing diagram and all associated text.

Plot Windows

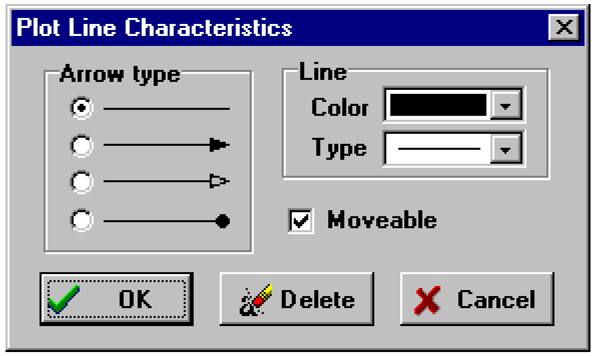
Variables which appear in the Parametric, Lookup or Array tables may be plotted with the New Plot Window or Overlay Plot commands in the Plot menu. In addition, plots of the thermodynamic properties can be generated using the Property Plot command. Up to 10 plot windows may be constructed, and each may have any number of overlayed plots. There are many plotting options such as choice of line type and plot symbol, linear/logarithmic scaling, spline fitting, tick frequency, and grid line control. These options can be set initially when the plot is first drawn or at a later time using the Plot window controls described below or the Modify Plot and Modify Axes commands in the Plot menu.

The appearance of the plot can be changed in many ways using the plot menu commands in the Plot menu and controls in the Plot window. The Plot window controls are as follows.

1. Moving the Plot
   The entire plot, including the axis scales and all text items, can be moved to a different location in the Plot window by holding the mouse button down in any location within the plot rectangle (but not on a text item) while sliding the mouse to its new location. An outline of the plot will move with the cursor and the plot will move to the new location when the button is released.

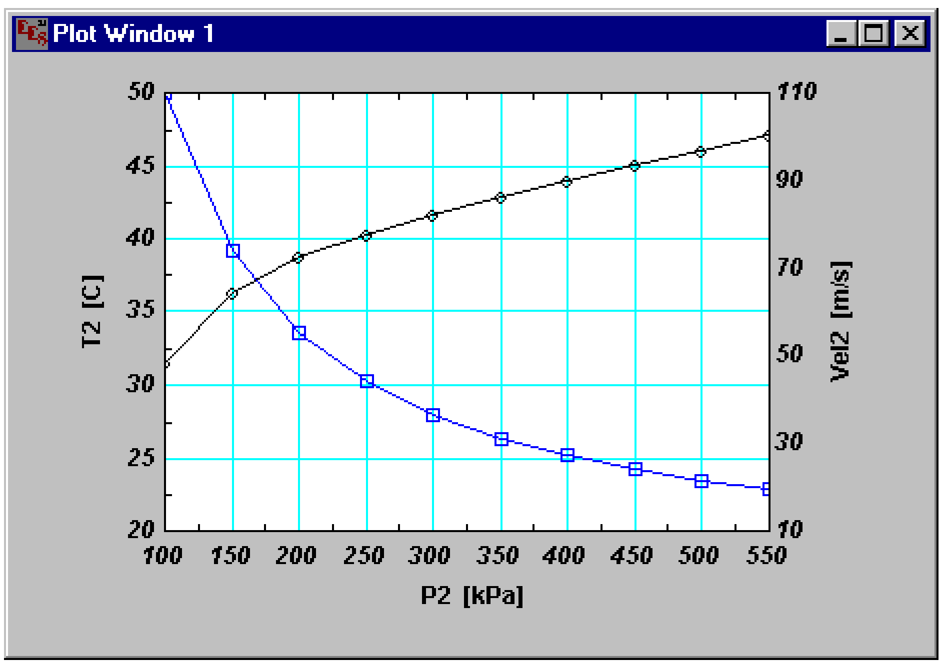
2. Moving Text
   Text items, such as the axis titles and any additional text you have added with the Add Text command in the Plot menu can be moved to any location within the Plot window by pressing and holding the left mouse button down while the cursor is on the text item and dragging it to its new location. A snap-to-grid option for text items is provided in the Plot Window tab of the Preferences dialog. When this option is selected, the text item will snap to the nearest position with the specified horizontal and vertical increments. Snap-to-grid can be overridden by holding the Ctrl key depressed as the text is moved.

3. Moving Lines and Arrows
   Lines and arrows can be placed on the plot using the Add Line command in the Plot menu. The choice of arrowhead on the line is made by double-clicking on the line which will bring up the following small dialog window. Select the desired type of arrowhead by clicking on the appropriate control. The line can be rotated or moved to a new location. To rotate the line, press and hold the left mouse button down while the cursor is positioned on either end of the line. The line will rotate to follow the cursor movement. Release the mouse button when the line is correctly positioned. To move the line to a new location, press and hold the left mouse button down while the cursor is over the center of the line and then drag the line to its new location and release the mouse button.



4. Resizing the Plot

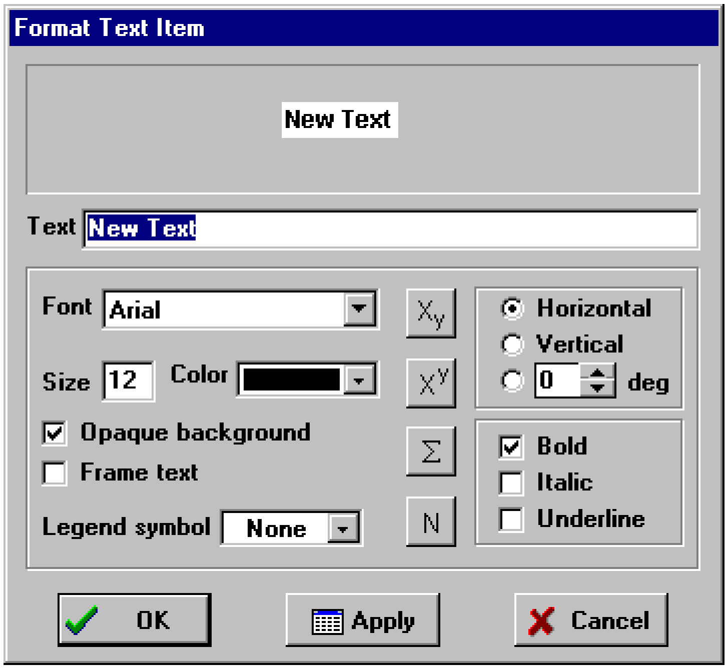
   The size or aspect ratio of the plot can be easily changed by pressing and holding the left mouse button with the cursor located at the lower right corner of the plot rectangle. The cursor will change from an arrow to the resize indicator (as shown below) when it passes over the resize control. The size of the plot will change as you drag the lower right corner to a new position. When the plot is resized, the size and positions of all text items and lines are proportionally changed.



5. Changing Text Characteristics

   The characteristics (e.g., font, size, style, color, orientation, etc.) of each text item can be individually changed by double-clicking the left mouse button while the cursor is positioned within the text rectangle. The Format Text Item dialog window shown below will appear displaying the text and its current characteristics. The text may be edited in the text edit field. Subscripts, superscripts, Greek or bold characters may be entered as follows. First select the text which is to be changed in the text box. Then click the X_y (subscript), X^y (superscript), Σ (Greek), or N (normal font) speed button. Control characters will be added to the text in the edit field. The text will be displayed as it will appear on the plot in the box at the top of the window.

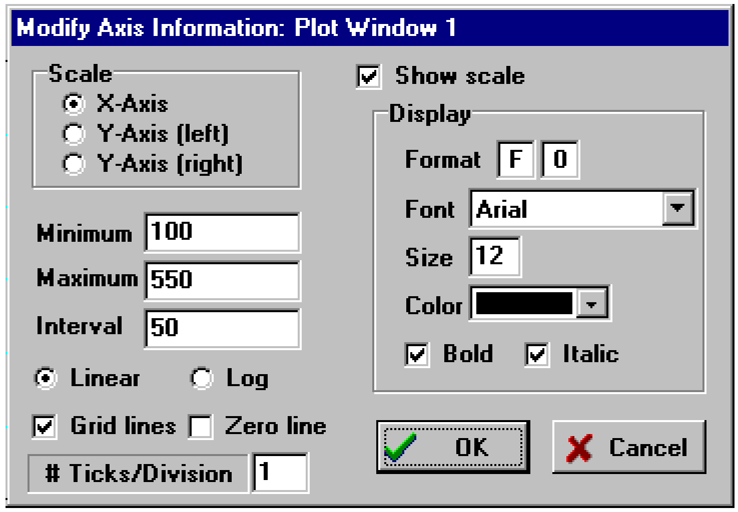
EES allows any horizontal text item to be associated with a plot symbol to facilitate construction of a legend. Clicking in the Legend symbol box will produce a drop-down list containing a descriptor of each existing plot. If a plot is selected, the line type and symbol used for that plot will be displayed just to the left of the text item and it will move when the text item is moved.



6. Modifying the Axis Information

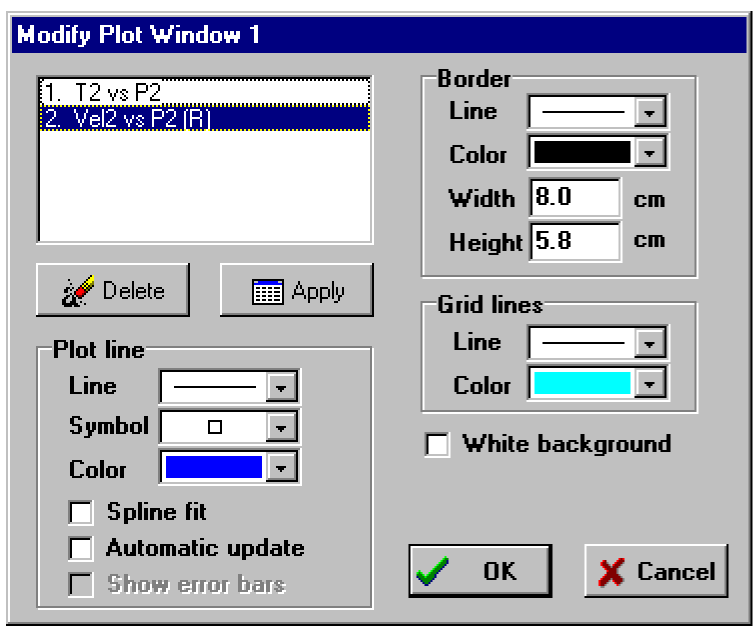
The axis scaling and appearance can be changed by double-clicking the left mouse button on the abscissa or (left or right) ordinate scales or by selecting the Modify Axes menu item in the Plot menu. Either action will bring up the Modify Axes dialog window.

The axis for which the changes are to be made is indicated by the radio-button controls at the upper left. The Minimum, Maximum, and Interval fields initially are the current values for the axis. These may be changed and the plot will be rescaled and redrawn. Scale numbers are placed at the position of each interval, as are Grid lines if selected. Selecting the Zero line causes a vertical (for x-axis) or horizontal (y-axis) line to be drawn at a value of zero. The No. Ticks/Division is the number of minor ticks, i.e., the number of tick marks between each interval. If Grid lines is selected, clicking on No. Ticks/Division will change it to #Grids/Division allowing grid lines to be placed at points in between the major ticks. If the Show Scale control is selected (as shown), the scale numbers will be displayed. The characteristics of these numbers are controlled by the remaining fields on the right-side of the dialog window.



7. Modifying the Plot Information

The line type, color, plot symbol (or bar type for bar plots), and other information relating to each plot can be viewed or modified by double-clicking the left mouse button anywhere within the plot rectangle (but not on text or a line.) The dialog window shown below will appear. This dialog window can also be made to appear with the Modify Plot menu item in the Plot menu. All current plots will be listed in the rectangle at the upper left in the order in which they were constructed. An (R) to the right of the plot name indicates that the plot uses the right-hand y-axis. Select the plot by clicking on its name.

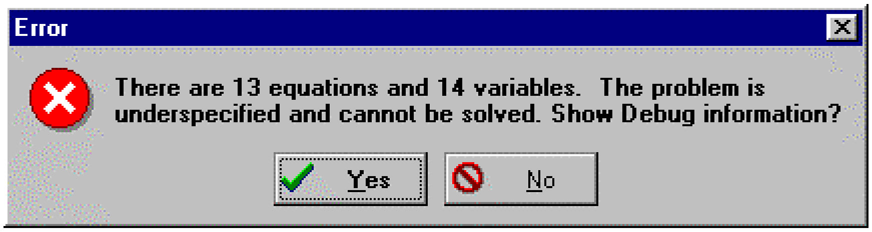


The Spline fit control, if selected, will cause EES to plot the line using cubic splines to produce a smooth curve through the data. Automatic Update sets up a direct link between the plot and the data in the Parametric Table so that the plot will be automatically redrawn if any change is made to the data in the Parametric Table. The

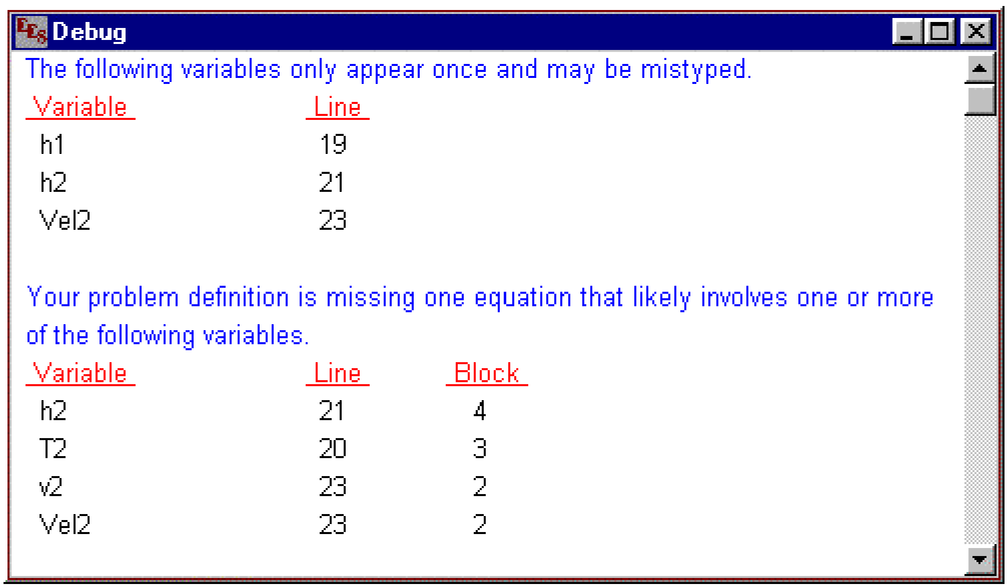
Show error bars control is enabled only if the data being plotted were obtained with the Uncertainty Propagation Table command in the Calculate menu. Click the Apply button if you wish to view the changes you have made.

Debug Window

The Debug window is a diagnostic tool that can be helpful in locating errors in your equations. Whenever an attempt is made to solve a set of equations in which the number of equations is not equal to the number of variables, a message box such as that shown below will appear.



Clicking the Yes button will bring up the Debug window. For example,



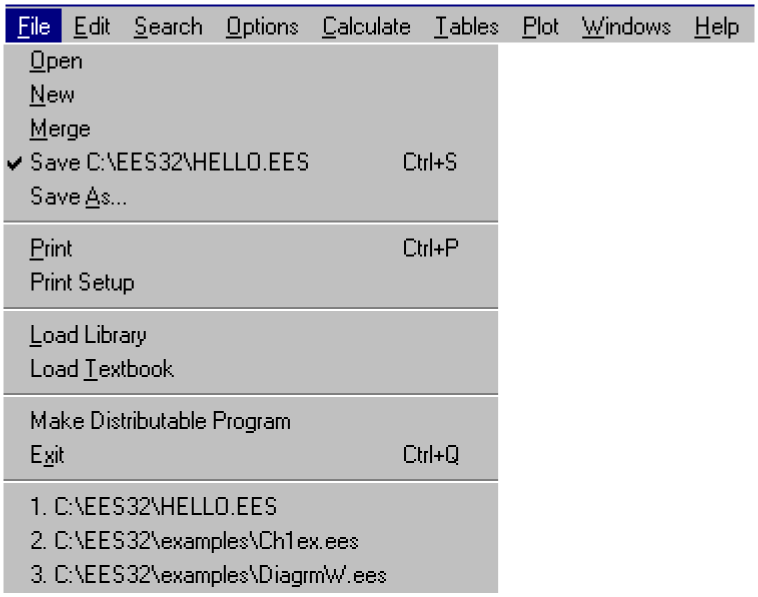
This window provides two lists of variables. The first list shows all variables which are referenced only once in the Equations Window. These variables are possibly spelled wrong or otherwise not being directly used in the problem, except for informational purposes. The second list shows the variables which are most likely to be involved in any missing or extra equations. The information used to construct this second list is determined by examination of the blocking order of the equations in the Residuals window. You may also find the information in the Residuals window helpful in identifying the problem with your equation set.

Clicking the left mouse button on a variable name in the Debug window will bring the Equations Window to the front with that variable name selected.

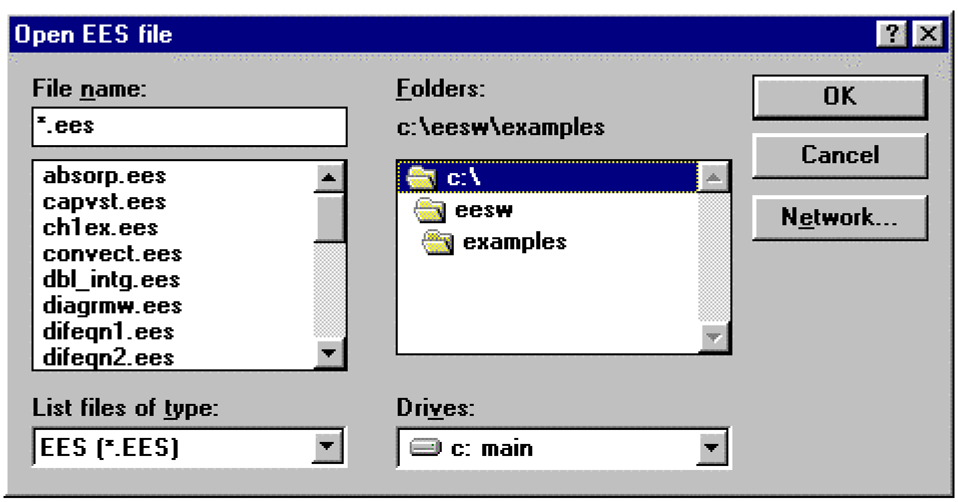
CHAPTER 3

Menu Commands

The File Menu



Open will allow you to access and continue working on any file previously saved with the Save or Save As… commands.



After the confirmation for unsaved work, the dialog window shown above will appear. The current directory is indicated in the Folders: field and EES files in that folder (directory) are shown in the list on the left. To select a file, click on the file name in the list or enter the filename in the File Name: field. You can open files in another directory by entering the directory name in File Name: field or by clicking on the folders listed in the Folders list. Clicking on the Drives list displays the available drive designations. Click on the drive name to select it. Choose the OK button to select the file (or directory) displayed in the Filename field.

EES can read four types of files which are identified as **EES file**, **Import file**, **Text file**, and **Library file** formats. The format is selected with the drop-down list at the bottom left of the dialog window. EES files with a .EES filename extension are the norm. Import files with a .XPT extension are files saved by EES with the Export option from a different operating system, such as the Macintosh. Text files with a .TXT filename extension contain ASCII text which is read into the Equations window. Library files are EES files containing one or more functions, procedures or modules which can be automatically loaded at startup, as described in Chapter 5.

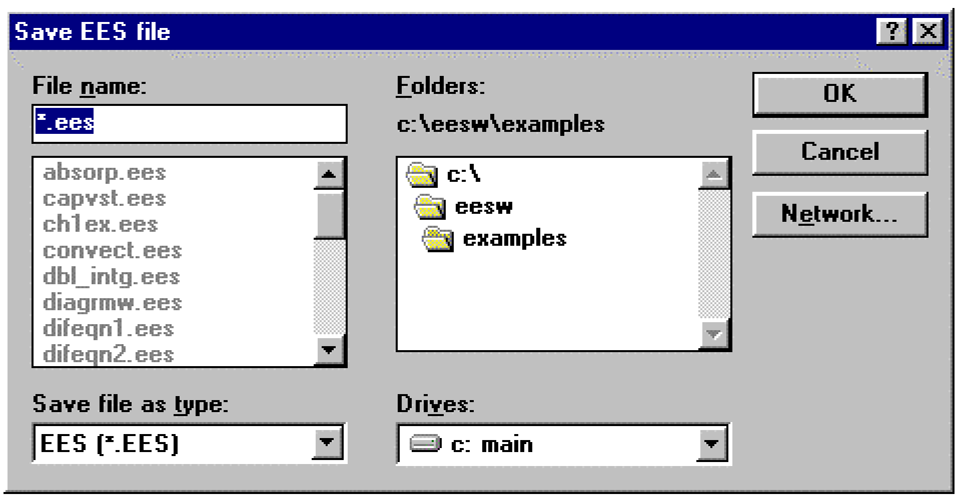
New initiates a new work session. All variables and equations will be cleared. The unit system will be restored to the settings that would be in effect if the program were restarted. If an unsaved problem definition exists, you will be asked if you first wish to save your current problem information.

Merge allows the equations previously saved in an .EES file to be merged with the current contents of the Equations window at the cursor position. The Merge dialog window operates in the same manner as for the Open command. Equations can also be entered from a text file using the $INCLUDE directive. EES functions, procedures and modules can be loaded using the Load Library command or $INCLUDE directive.

Save will save your problem definition with the same file name (which appears after Save in the File menu and in the title bar of the Equations window) as it was last saved. For a new work session which has not yet been named, you will be prompted to supply a file name, just as if the Save As… command were given. All information concerning the problem definition is saved, including the equations, variable information, the tables, the plots, and the size and locations of the windows. By default, the file will be saved in the standard EES file format with a .EES filename extension. If you wish to export the file to a version of EES on a different operating system, use the Export format in the file type available in the Save As… command. A check mark will appear to the left of the word Save in the File menu if the current problem information has been saved on disk. Any change in the problem information will cause the check mark to disappear.

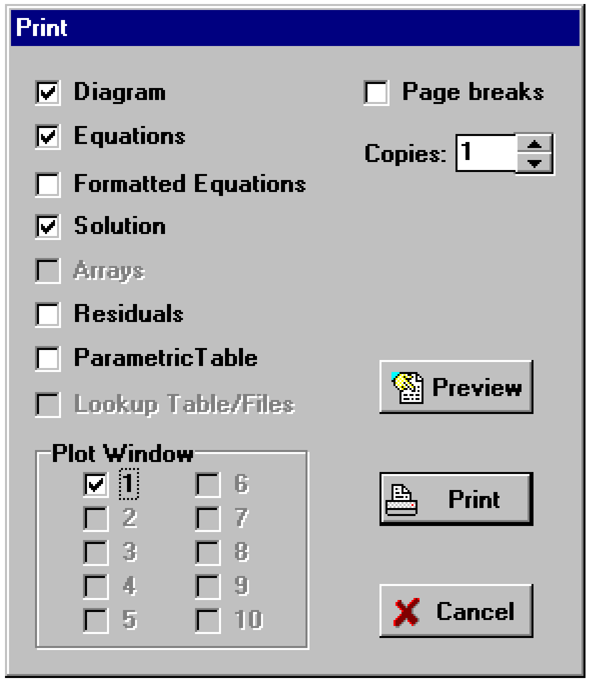
Save As… provides the same function as the Save command except that it will first prompt you to supply a filename in the Save File dialog window. The Save As command allows the problem definition to be saved with another filename or in a form which can be exported to EES versions on other operating systems. Enter the file name of your choice in its place. The 32-bit version of EES supports long file names. The file name may include drive and directory information. However, it is not necessary to enter a filename extension, since EES will supply the extension automatically.

EES recognizes four file types. If EES is displayed in the Type box at the lower left, the extension in the File Name: field will be set to .EES (the norm) and files having this extension will be displayed in the filenames list. The Export file type will apply a .XPT filename extension and save the file in a generic ASCII format which can be transferred to other operating systems such as the Macintosh. The Text file type will apply a .TXT filename extension and save only the text in the Equations window in an ASCII file. The Library type will change the filename extension to be .LIB. Each time EES is started, it opens all of the .LIB files in the USERLIB\ sub-directory and automatically loads the functions, procedures, and modules in these files. These functions can be used exactly like the EES built-in functions. Library files are one of the most powerful features of EES because the user can easily develop special purpose functions. See Chapter 5 for additional information.



Print will print any or all of the EES windows to the printer or to a file on the disk. Each window has a small check box preceding its name. If the check box is grayed (as it is for the Arrays window in the Print dialog window shown below) the window is not available for printing. If an X appears in the box, the window will be printed. To place an X in the box or to remove an existing X, click the mouse while the cursor is positioned in the box. If the Page breaks check box is selected, a forced page break will occur so that the printed output for each window starts on a new page. The printed output will be sent to the default printer selected with the Printers application in the Windows Control Panel group.

It is possible to direct the output to a file, rather than a printer, with the Connect options in the Printers applications. See the Windows manual for additional information on selecting printers. Printing options such as the font, line spacing, and font size are set in the Preferences dialog window (Options menu). The Preview button will direct a facsimile of the printed output to the screen.

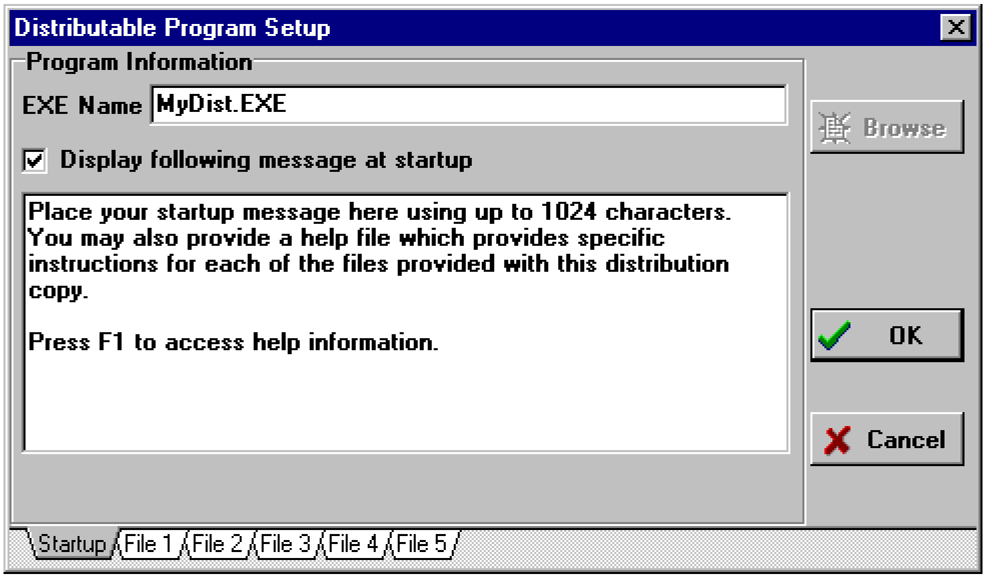


Printer Setup provides a dialog window which allows the printer to be selected along with printing options, such as the paper size and orientation.

Load Library will bring up the standard open file dialog showing EES Library files (which have a .LIB file name extension) in the file selection box. Library files contain user-supplied functions, procedures, and/or modules which operate as described in Chapter 5. Once loaded, these library files remain in memory until EES is closed. Note that when EES starts, it will preload all of the library and externally-compiled files which are found in the USERLIB\ sub-directory so the Load Library command is not needed for these files. The Load Library can also be used to load external functions and procedures with filename extensions of .DLF, .DLP, and .FDL. See Chapter 6 for additional information. The $INCLUDE directive described in Chapter 7 can also be used to load library files.

Load Textbook reads a user-generated Textbook index file with the filename extension (.TXB) and uses the information in this file to create a Textbook menu at the far right of the menu bar. A textbook index file can also be loaded automatically, by placing the textbook index file and associated problem files in a subdirectory within the USERLIB subdirectory The Textbook menu provides convenient access to set of problems, such as those developed for use with a text. The Textbook menu section at the end of this chapter provides instructions for creating and using a Textbook menu.

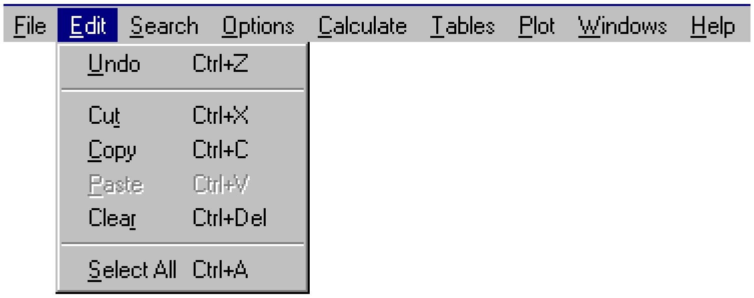
Make Distributable Program will create a special purpose version of EES which will run one to five pre-selected problems. The following dialog will appear when this menu command is chosen. A special version of the EES program with one to five EES problems and all supporting files are placed in a single executable file when you select the OK button. This executable file can be freely distributed to others.



Quit provides a graceful way to exit the program.

The remaining items in the File menu are recently accessed filenames. Selecting any of these filenames opens the file. This list can be disabled in the Preferences dialog. It is recommended that this list be disabled if the program is installed and used on a network.

The Edit Menu



Undo restores the Equations window to the condition it was in before the last editing operation. The Undo command is available only for the Equations window.

Cut deletes the selected (highlighted) text. The deleted text is placed on the Clipboard where it can be pasted to another location with the Paste command.

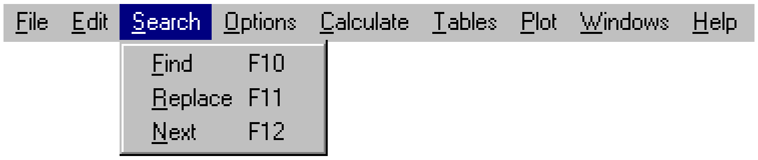
Copy functions in a manner dependent on which window is foremost. Copy will place the selected text from the Equations window on the Clipboard from which it can be restored at the cursor position with the Paste command. When the Parametric, Lookup, or Array Table window is foremost, the Copy command will copy the selected cells (shown in inverse video.) The data copied from the table are stored on the Clipboard in a standard format in which numbers within the same row are separated by a tab and each row ends with a carriage return - line feed. Data in this standard format can be pasted into any location of the Parametric or Lookup tables or into other applications. (Hold the Ctrl key down when selecting the Copy command if you wish to also copy the column heading and units.) Copy will move a selected Plot window or the Diagram window graphics into the Clipboard from which it can be pasted into other applications. If you hold the Ctrl key depressed while copying the Diagram window, the text items added with the Add Diagram Text command will not be included. The copy of the Plot or Diagram window is stored in MetaFilePict object format. Copy Solution will place the contents of the Solution window on the Clipboard both as ASCII text with each variable is on a separate line and as a picture of the formatted Solution window. Use the Paste Special in an another application, such as a word processor, to select either the text or the picture for pasting. See the description of the Solution Window for more information. Copy will place the entire contents of the Residuals window on the Clipboard as text. Tabs separate the different items on each line of the Residuals windows. The Clipboard contents can be pasted into a word-processor.

Paste is active for the Equations, Parametric, Lookup, and Diagram windows. Paste moves the text (or graphics for the Diagram window) previously placed on the Clipboard with the Cut or Copy commands in EES or in other applications. When Paste is used in the Parametric or Lookup Table window, the values stored on the clipboard will be copied to the table starting in the cell in which the cursor is currently located. Data can thus be moved between the Parametric and Lookup tables. Copy and Paste can also be used with text items placed on the Plot windows.

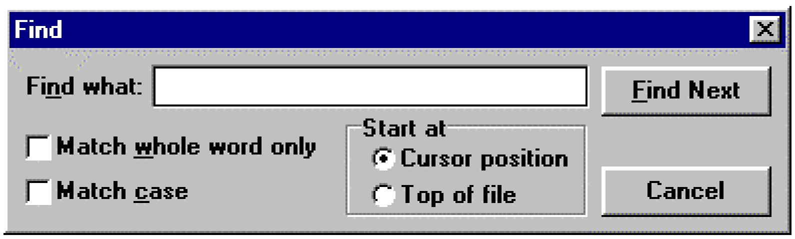
Clear removes the selected text without placing a copy on the Clipboard. Clear can also be used to delete the contents of the Diagram window.

Select All will select all of the text in the Equations window, or all of the cells in any of the three tables, depending on which window is foremost when the command is issued. The Select All command is normally followed by Copy which places the selected items on the clipboard. If the Formatted Equations window or Solution window is foremost, the Select All command appears as Select Display. This command will select all currently visible items in the window.

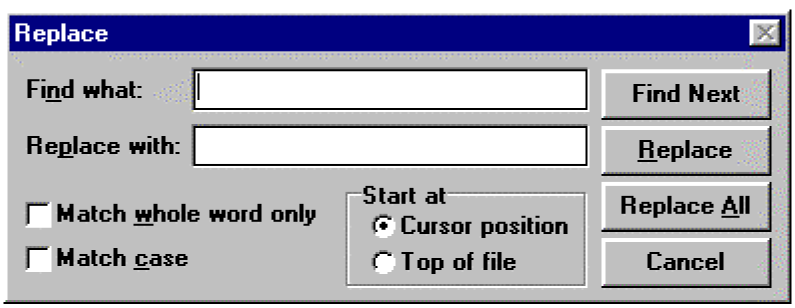
The Search Menu



Find will search the Equations window for the first occurrence of the text entered in the Find what: field. The search is case-insensitive unless the ‘Match case’ option is selected. If the ‘Match whole word only’ option is selected, the text will be found only if it is delimited by spaces or mathematical operators. The Cancel button will change to Done after the find process is completed.

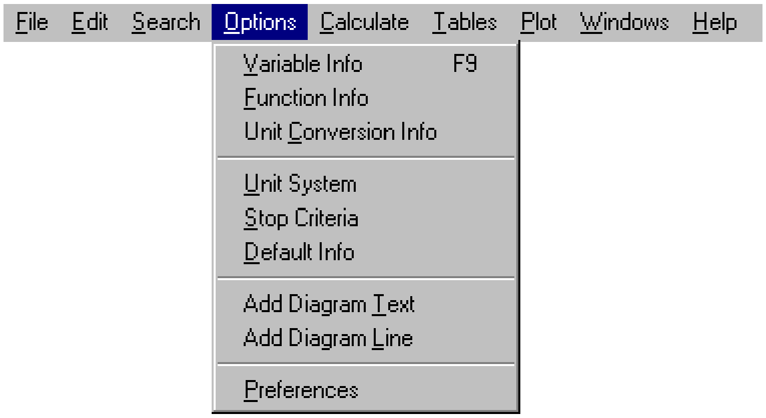


Replace will search the Equations window for the first occurrence of the text in the Find what: field and replace it with the text in the Replace with: field. The search options are as described for the Find command. The Replace All button will replace every occurrence of the search text with the replacement text. The Cancel button will change to Done after the find process is completed.

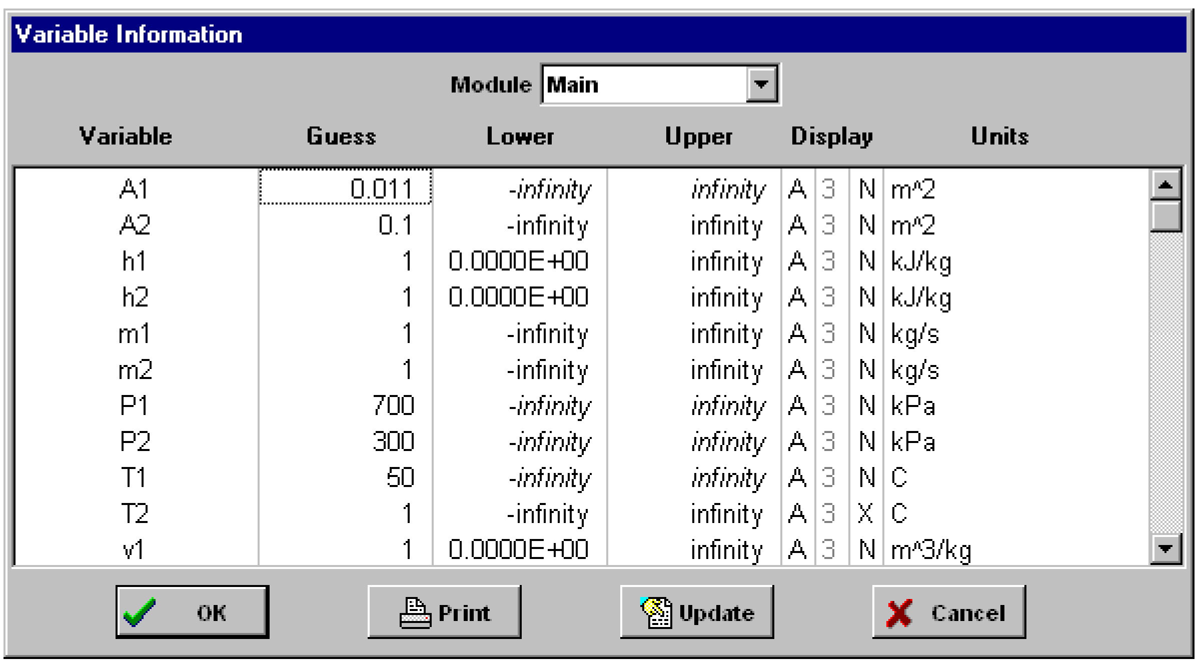


Next will find the next occurrence of the text previously entered with the Find or Replace command. The search options will be remain in effect if they were set in the Find command.

The Options Menu



Variable Info will provide a dialog window, as shown below, in which the guess value, lower and upper bounds, display format, and units of all variables currently appearing in the Equations window can be viewed or changed. These data are initially set to default values. The defaults, selected based on the first letter of the variable name, may be set with the Default Info command.



If the program contains one or more modules (see Chapter 5 for a discussion of modules), a control will be provided at the top of the dialog to select the module or main program. The information for the selected module can then be changed or viewed.

Use the scroll bar on the right side of the dialog window to bring the variable information into view. Note that the height and width of this dialog window can be changed by selecting an edge and dragging it in the usual Windows manner. All fields, including the variable name, may be changed as required. If the variable name is changed, EES will change every occurrence of the original variable name in the Equations, Parametric table and Diagram windows.

The words, -infinity and infinity can be used to indicate unlimited lower and upper bounds, respectively. The Guess, Lower and Upper value fields will also accept a variable name, as well as a number. When a variable name is provided, EES uses the current value of that variable as the guess value or bound.

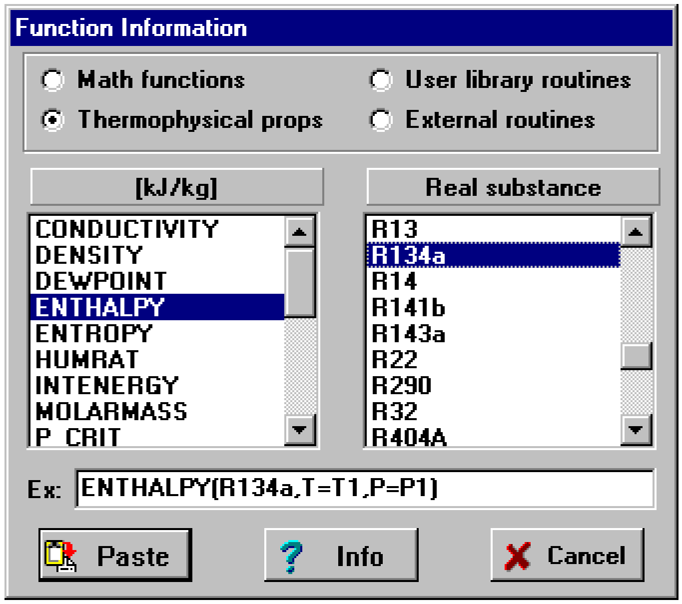
EES attempts to solve equations having a single unknown before this display appears. Variables for which the value has been pre-calculated are identified by having the bounds shown in italics. The pre-calculated value appears in the Guess column. These guess values and bounds may still be edited which will then cause EES to recalculate the value.

The display format of a variable in the Solutions or Table window is controlled by the three fields in the Display columns. Clicking in these fields will produce a pop-up menu for the display style, number of significant digits, and hilighting effects.

The units of the variable (or any other desired information) may be entered in the units column. Units are used by EES only for display purposes in the Solution and Parametric Table windows. Note that the display format and units of each variable can also be changed by clicking on the variable in the Solution Window.

When the OK button is pressed, *all* changes made to the variable information since the dialog window appeared are accepted. The Update button replaces the guess value of each variable with its current value, i.e., the value determined in the last calculation. The same update feature is provided with the Update Guesses command in the Calculate menu. The Print button will direct a copy of the information in this table to the selected printer. The Cancel button will restore all fields to the condition they were in when the Variable Information dialog was first presented and then remove the dialog.

Function Info will bring up the following dialog window.

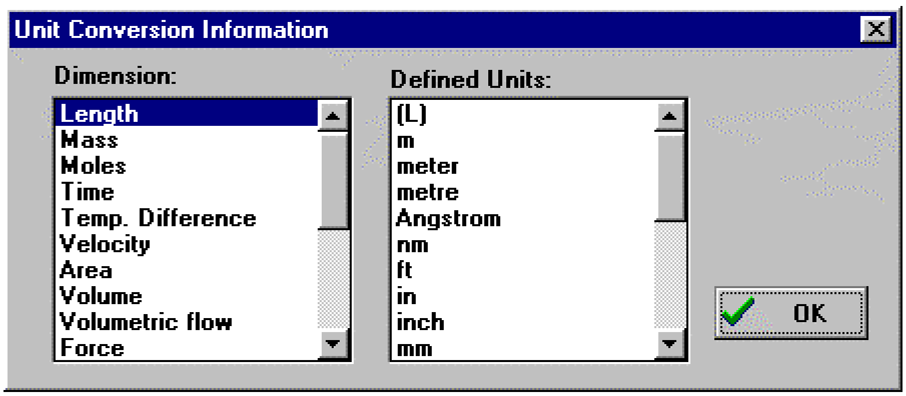


The four buttons at the top of the dialog window indicate which information is to be provided. Math Functions and Thermophysical Props refer to the built-in functions for mathematical and thermophysical property relations, respectively. The User Library button provides a list of the user functions, procedures, and modules loaded from Library files. (See Chapter 5 for additional information on Library files). The External Routines button refers to compiled routines which can be linked to EES as described in Chapter 6. The functions corresponding to the selected button will be displayed in the Function list on the left. To select a function, click on the function name in the scrollable list. Click the **Info** button at the bottom of the dialog to obtain specific information relating to the function you have selected.

The units of the thermophysical property function are shown above the function list box. Thermophysical property functions require specification of a substance. The substances for which property data are available are shown in the Substance list on the right. Click on the substance name in the scrollable list to select the substance. 'Ideal gas' will appear above the substance list if the properties of the selected substance are calculated using ideal gas law approximations. 'Real substance' will appear if liquid and vapor states are determined. Substances represented by their chemical formula (e.g., CO2) are modeled as an ideal gas and use JANAF table reference values for enthalpy and entropy. Substances with their name spelled out (e.g., CarbonDioxide) are modeled as real fluids and do not use JANAF table reference values. Air is an exception to this rule. Air is modeled as an ideal gas. Psychrometric functions are applicable only to the substance AirH20. Additional information regarding all built-in functions is provided in Chapter 4.

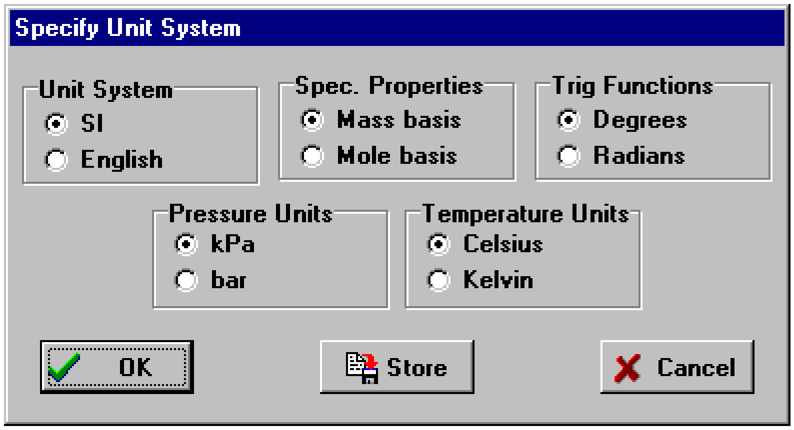
An example of the function with default variables will be shown in the Example rectangle at the bottom. You can edit this information in the usual manner. If you click the Paste button, the contents of the Example rectangle will be pasted into the Equations window at the cursor position.

Unit Conversion Info provides information to support the use of the Convert unit conversion function. The Convert function has the following format: Convert('From', 'To') where From and To are character strings identifying the unit type such as 'Btu/hr-ft2-R' or 'mph'. (Note that the single quotes marks around the unit identifies are optional.) Many of the unit identifiers are obvious, but not all. The purpose of this command is to list the unit identifiers that have been defined, as shown below.

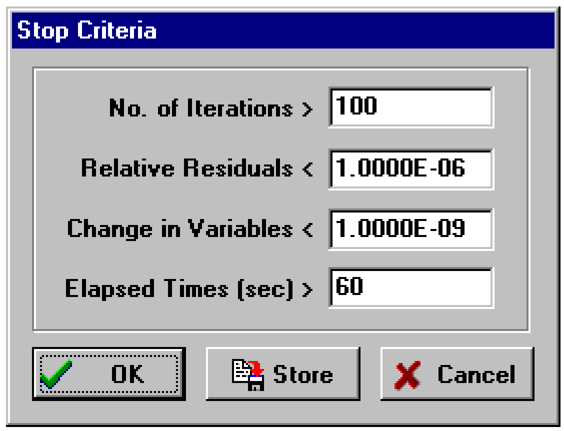


Click on the dimension in the list at the left. All of the units which have been defined with that dimension are listed in the list on the right. Note that only the defined units having the selected dimension are listed. For example, if you click on Area, only Acre and Hectares will be displayed. However, any combination of units having the dimensions indicated at the top of the right list (e.g., L^2 for Area) can be used in the Convert function. You can add additional units if needed. They are stored in the UNITS.TXT file in the main EES directory. Instructions for adding information are provided at the top of the file. This file may be read as a text file and edited in EES.

Unit System provides a dialog window shown below in which the units of the variables used in the built-in mathematical and thermophysical property functions may be set. The unit settings are displayed in the Solution window. The unit system is only needed for the built-in function calls. EES does not provide automatic unit conversion. The units will be changed for the remainder of the work session if the OK button is pressed. The selected units are saved with other problem information when the Save command in the File menu is issued. These units are then restored with the problem using the Open command. If you wish to permanently change the default values, press the Store button.



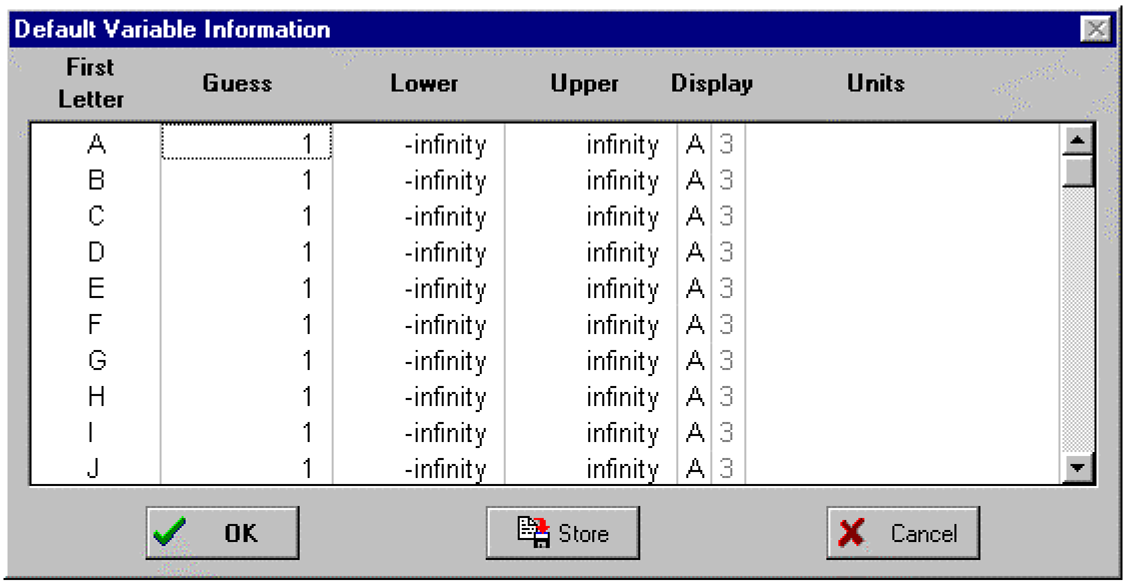
Stop Criteria allows the specification of criteria which will terminate the iterative solution.



The criteria are the number of iterations, the maximum relative residual, the maximum change in a variable value from one iteration to the next and the elapsed time. If any of these criteria is satisfied, the calculations terminate. All calculations in EES are done in extended precision with 21 digits of significance. Loss of precision is unlikely to be a problem even when very small values are set for the maximum residual or variable change. However, small values for these quantities increase the number of iterations required for a solution and therefore the computation time. The stopping criteria will be set as displayed for the remainder of the session by pressing the OK button. The stopping criteria are saved with other problem information when the Save command in the File menu is issued and restored using the Open command. To change the default stopping criteria that EES presents at the start of a new session, press the Store button.

Default Info provides a means for specifying the default guess values, bounds, display format and units of new or existing variables depending on the first letter in the variable name. There are two ways to use this command. If the problems you do all tend to have the same nomenclature, it is best to set the default variable information and save it by pressing the Store button. The Store button will cause the current default settings to be permanently saved so that these defaults will appear at the start of the program the next time EES is run.

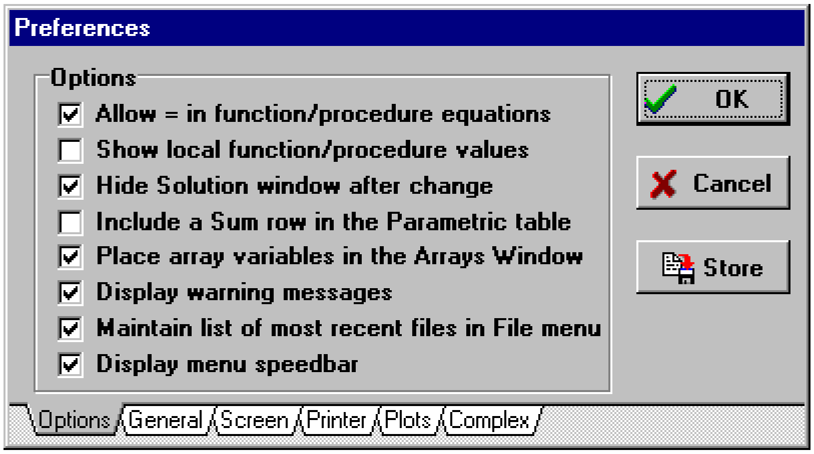
The Default Variable Information command can also be used to selectively change information for existing variables. For example, if you change the units for variables beginning with letter T to [K] and press the OK button, all existing variables beginning with letter T will take on these new units. No other changes to existing variables will be made. Each new variable beginning with letter T will then also take on units of [K]. The OK button sets the current default setting for this problem session only.



Add Diagram Text provides a dialog window in which text of three types may be placed on the Diagram Window. The three types are plain text, input variables, and output variables. An input variable provides an edit box in which the value of a variable can be entered. An output variable displays the most recently calculated value of a selected EES variable on the Diagram window. See the Diagram Window section of Chapter 2 for details.

Add Diagram Line allows lines or arrows to be placed on the Diagram Window. The command functions in exactly the same manner as the Add Line command in the Plot menu except that the line is drawn on the Diagram Window rather than on a Plot window.

Preferences provides six tabs for user choices concerning program options, general display options, screen display, printer display, plot window, and complex number options. These options are shown and described below. If the OK button is clicked, the selected preferences remain in effect for the remainder of the work session. The Store button saves the preferences so that they will be in effect at the start of the program the next time EES is run.



☒ **Allow = in function/procedure equations** will suppress the error message that would normally occur if the assignment symbol (:=) is not used in EES Functions and Procedures. EES Internal Functions and Procedures (Chapter 5) employ assignment statements as in FORTRAN and Pascal, rather than equations as used in the main body of the EES programs. (EES modules use equality statements as in the main body of the EES program and thus cannot use the := assignment statement syntax.) An assignment statement sets the variable identified on the left of the statement to the numerical value on the right. X:=X+1 is a valid assignment statement, but it obviously is not an equality. The := sign is used to signify assignment, but if this control is selected, EES will also accept X=X+1.

☒ **Show function/procedure/module values** will allow the most recent values of local variables in EES functions, procedures and modules to be displayed in the Solution window. Module equations will also appear in the Residuals window. The values of these local variables are ordinarily not of interest but you may wish to know them, particularly for debugging purposes. Note that only the functions, procedures, and modules which appear in the Equations window are affected by this setting. The local values of variables in Functions, procedures, and modules that have been loaded from library files (See Chapter 5) are not displayed.

☒ **Hide Solution Window after change** causes the Solution, Arrays, and Residual windows to be removed from the screen display if a change is made in the Equations window. If this option is not selected and a change is made in the Equations window, the Solution window title will change to Last Solution.

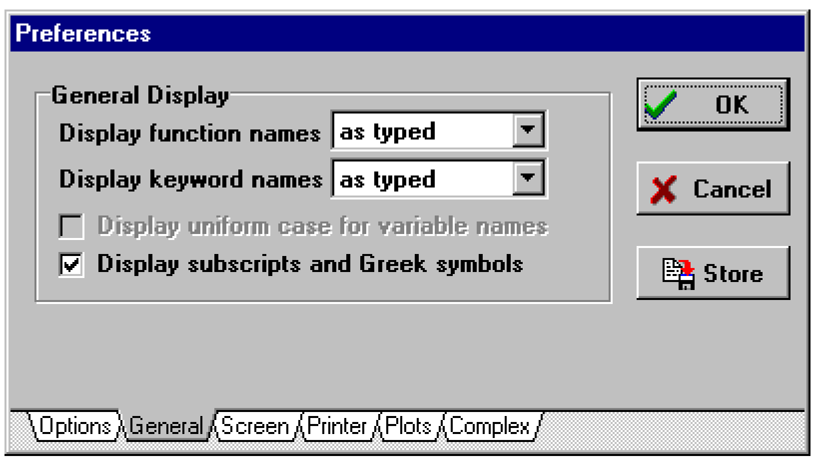
☒ **Include a Sum row in Parametric Table** will result in an extra row being added to the Parametric table which displays the sum of the values in each column.

☒ **Place array variables in the Arrays Window** instructs EES to display all array variables in the Arrays window rather than in the Solution window after calculations are completed. Values in the Arrays window can be plotted and copied just like values in the Parametric and Lookup tables. See the Arrays Window section of Chapter 2 for additional information. The existing Arrays window is destroyed if this option is unselected.

☒ **Display warning messages** will enable or disable warning messages during calculations. Warnings are issued if thermophysical property correlations are applied outside of their range of applicability. Warnings can also be turned on or off using the $Warnings On/Off directive in the Equations window.

☒ **Maintain a list of recent files in the File menu** enables or disables a list of up to 8 recent files at the bottom of the File menu. This list is a convenience that you normally would want to have. The file names are stored in a file entitled EES.FNL in the EES directory. However, if EES is placed on a server where multiple users can access the program, it is best to disable this feature.

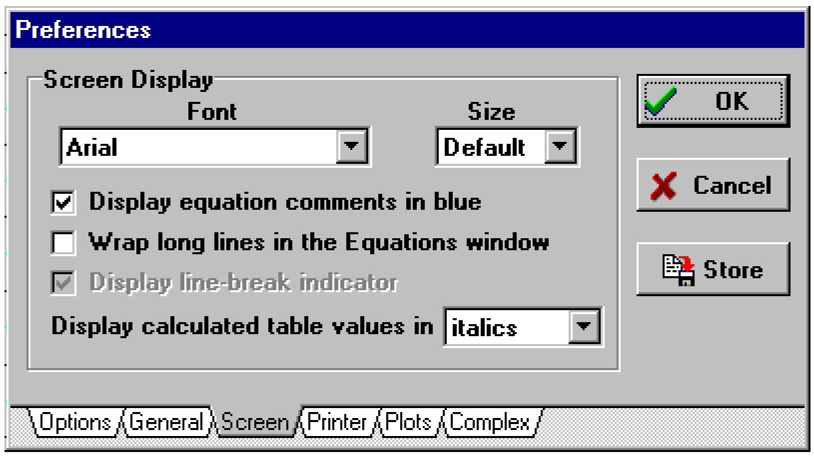
☒ **Display menu speedbar** controls the visibility of the toolbar appearing below the menu bar. The toolbar will be hidden if this control is unselected.



The first two options under the General tab allow the function names (such as ENTHALPY, SIN, etc.) and keywords (such as FUNCTION, DUPLICATE, fluid names, etc.) to be displayed in upper case, lower case, or as typed.

☒ **Display uniform case for variable names** causes each variable to appear with the upper and lower case lettering sequence set in the first occurrence of the variable in the Equations window. If the first occurrence of the variable is changed, the Check/Format command in the Calculate menu will change all other occurrences.

☒ **Display subscripts and Greek symbols** affects the appearance of EES variables in the Solution and Formatted Equations windows. When this option is selected, array variables will be displayed with the array index as a subscript[6].. The characters following an underscore in a variable name will also normally be displayed as a subscript, except for _dot, _bar, and _infinity. The _dot and _bar designation places a dot or bar centered over the variable name The _infinity will provide an infinity symbol (∞) subscript. Variables names which are part of the Greek alphabet (e.g., alpha, beta, gamma, etc.) will be displayed in symbol font. If the entire variable name is in capital letters, an upper case Greek symbol will be used; otherwise lowercase will be used.



Changing the Font and Size in the Screen tab will change the appearance of all EES windows except for the Plot and Diagram windows which have separate controls for text formatting.

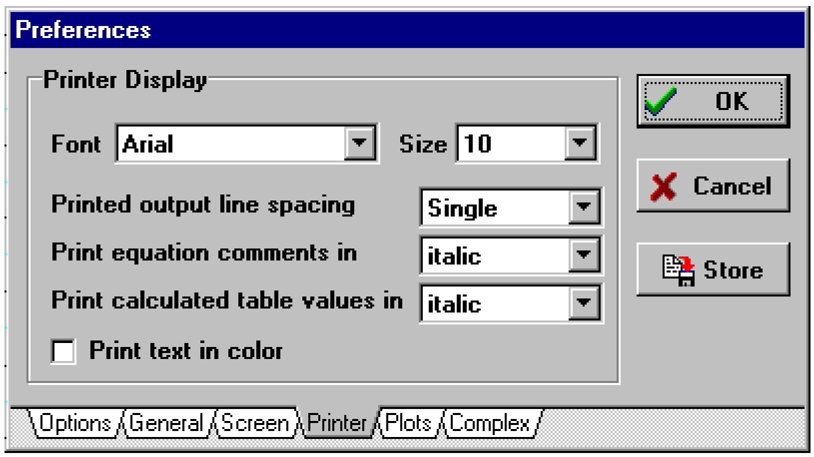
☒ **Display equation comments in blue** will cause EES to display comments, i.e., text between {curly braces} or “quote marks” in blue on color monitors. Note that this option will necessarily reduce the speed in which the Equations window can be displayed. You may wish to disable this option when working with large files on slow machines.

☒ **Wrap long lines in the Equations window** will hide the horizontal scroll bar. Lines which are too long to be displayed in the Equations window will be broken at an appropriate point and continued on the following line. A red > symbol will be displayed in the left margin of continuation lines if Display line-break indicator is selected.

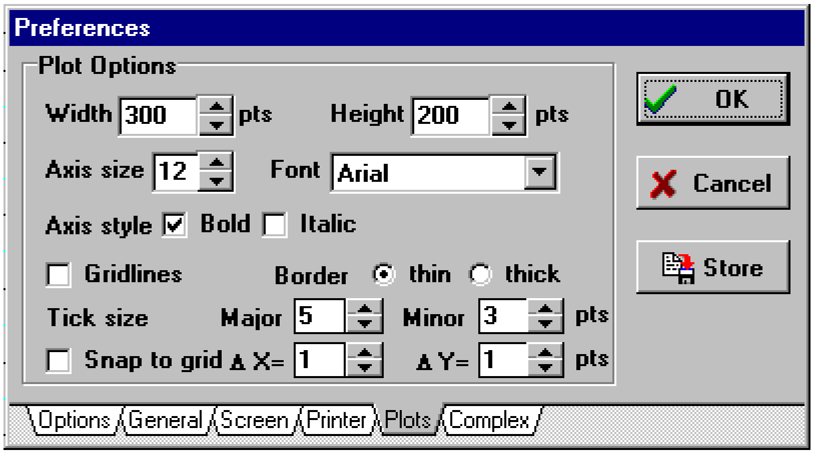
[6] Although array variables, e.g., A[1], and variables containing an underscore, e.g., A_1, will both display as A_1 in the Formatted Equations window, they are different variables and they have different characteristics. The index of array variables can be used within the scope of Duplicate statements, or with the Sum and Product functions. In addition, calculated values of array variables can be displayed in the Arrays window and they can be plotted.

☒ **Display line-break indicator** is applicable only if the Wrap long lines option is selected. This option controls whether line break characters appear in the left margin of continuation lines.

Display calculated table values in (same font, italics, blue, bold) controls the appearance of cells in the Parametric table which have been calculated by EES. Values entered by the user are always displayed in normal style in the selected font and font size.

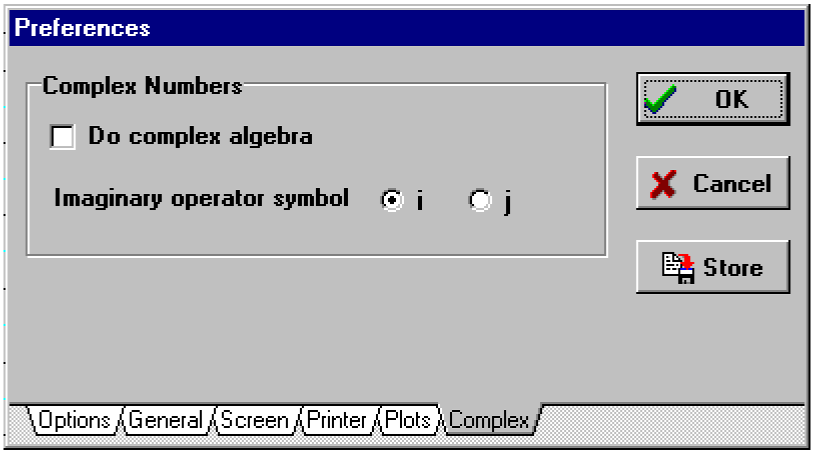


The options in the Printer Tab only affect the appearance of the printed output. Colored text does not appear clearly on some printers. If the ☒ Print text in color option is off, all text will be converted to black. The Printed output can be viewed using Print Preview in the Printer dialog.



The Plots tab allows the default setting for the plot width and height, the font, font size, font style, and the major and minor tick sizes for the axis scales to be changed. The plot width and height are entered in point units. Depending on your equipment and video setting, a point is either 1/96 or 1/120 inch. Ticks are the short line segments on the axis scale. Major ticks are placed on the scale at the point where the axis numbers appear whereas minor ticks occur between the axis numbers. Ticks which are drawn into the plot rectangle are represented with positive numbers. The plot can be configured for outdented ticks by specifying negative values for the tick sizes. These default characteristics are applied whenever a new plot is generated.

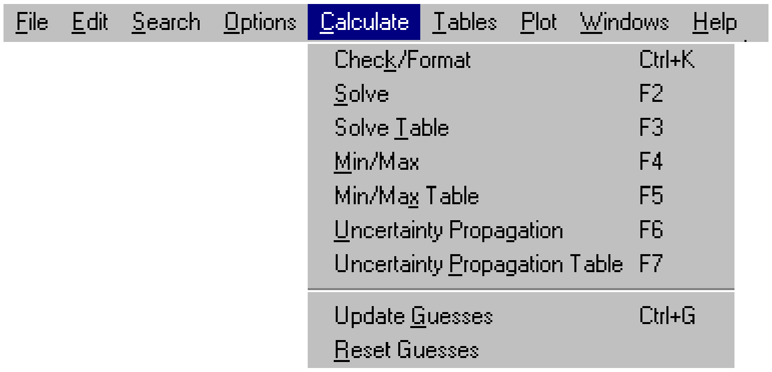
The snap to grid will be in effect if the a check appears in the ☒ Snap to grid control. The spacing of the grids in the horizontal and vertical directions is entered with the spin controls. The number appearing in these controls is the snap grid spacing. For example, a value of 4 will cause the text to be aligned to the nearest pixel divisible by 4. When this option is enabled, it is easier to line up text options used in forming a plot legend. Pressing the Ctrl key while moving text will disable the snap to grid option.



The Complex Tab allows the complex algebra capability in EES to be turned on or off. The complex capability can also be turned on or off with the $COMPLEX ON/OFF directive. The imaginary variable name representing the square root of –1 can be designated to be either i or j depending on the radio button setting.

Pressing the OK button will set the Preferences for this session only. The Store button will cause the Preferences to be permanently saved so that they will be in effect at the start of the program the next time EES is run.

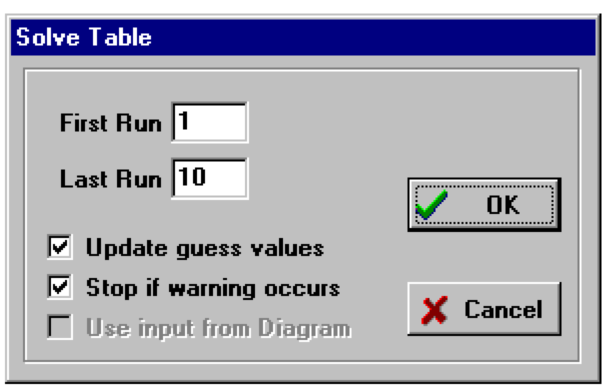
The Calculate Menu



Check/Format will recompile all equations and apply the formatting options selected with the Preferences command in the Options menu. The first syntax error found will be indicated with a message. If no syntax errors are encountered, EES will indicate the number of equations and variables in the Equations window.

Solve will first check the syntax of the equations in the Equations window. If no errors are found and if the number of equations is equal to the number of variables, a solution to the equation set will be attempted. The methods used by EES for solving equations are described in Appendix B. An information dialog window summarizes the progress of the solution. When the calculations are completed, the information dialog indicates the elapsed time, number of blocks, the maximum residual (i.e., difference between the left and right hand sides of an equation), and the maximum change in the value of a variable since the previous iteration. If the number of equations is not equal to the number of variables, EES will provide the option of viewing the Debug window which may help locate a problem in the problem definition. If the Diagram window is being used to enter one or more variable values, it must be open when the Solve command is issued. See Chapter 2 for more information concerning the Debug and Diagram windows. I

Solve Table will initiate the calculations using values specified in the Parametric Table. (See the description of Parametric menu commands on the following pages for information on the use of the Parametric Table.) The following window will appear.

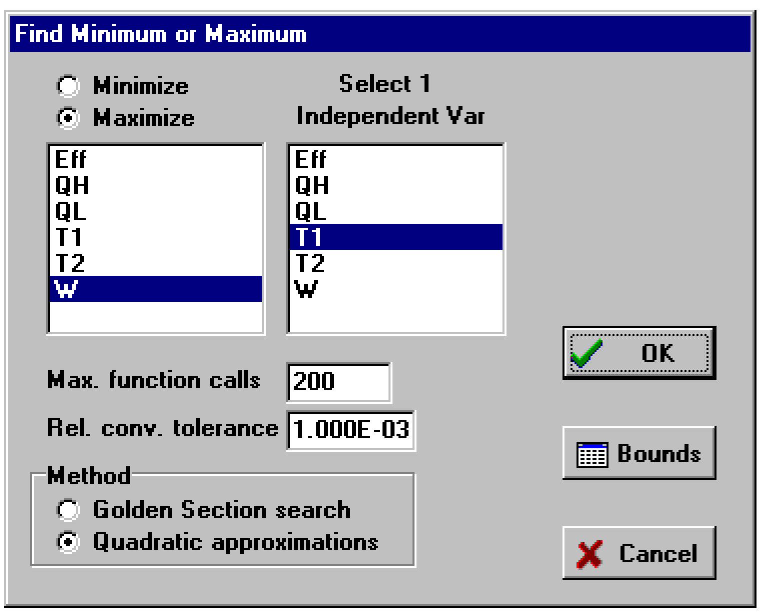


Each row of the table is a separate problem. The values of independent variables are shown in normal type. Blank cells (or cells with bold, blue, or italic type from a previous Solve Table command) are dependent variables. The values of these variables will be cleared and the newly calculated values will be entered in the table. If the Update Guess Values control is selected, the guess values for each run will be set to the calculated values from the previous run; otherwise each run will be initiated with the guess values specified with the Variable Info command.

Warnings are issued if a property correlation is applied outside of the range for which it was developed. If the Stop if warning occurs control is selected, EES will terminate the table calculations on the run at which the warning occurred. otherwise, EES will continue the calculations for the remaining rows. A message will be displayed after all of the table calculations are completed indicating the rows in which a warning occurred.

The "Use input from Diagram" control will be enabled if the Diagram window is open and has one or more input variables. If this control is selected, EES will accept input values from the Diagram window, just as if these inputs were specified in the Equations window.

Min/Max is used to find the minimum or maximum of an undetermined variable in an equation set for which there is one or more (limited to 10) degrees of freedom. EES will first check the syntax of the equations in the Equations window. If no errors are found, a dialog window will appear presenting the undetermined variables in two lists.



Click the Minimize or Maximize button above the left list. The variable which is to be minimized/maximized is selected by clicking on its name in the list on the left. The independent variable(s) whose value(s) will be changed in searching for the optimum appear in the list on the right. It is necessary to select as many independent variables as there are degrees of freedom in the Equations window. The number of independent variables which must be selected is indicated above the right-hand list. To select (or unselect) a variable, click on its name in the list.

If there is one degree of freedom, EES will minimize/maximize the selected variable using either a Golden Section search or a recursive quadratic approximation method, depending on the settings of the buttons at the bottom of the dialog window. (See Appendix B for information on the optimization algorithms.) The recursive Quadratic Approximations method is usually faster, but the Golden Section method is more reliable. Multi-dimensional optimization may be done using either Direct Search or a Variable Metric algorithm. The Variable Metric method, which uses numerical derivatives, usually performs much better than the Direct Search method, but it may be confounded if the optimum is constrained to be on a bound.

EES requires finite lower and upper bounds to be set for each independent variable. Careful selection of the bounds and the guess value(s) of the independent variables will improve the likelihood of finding an optimum. You can view or change the bounds and guess value for each selected independent variable by clicking the Bounds button. This will bring up an abbreviated version of the Variable Info dialog containing just the selected independent variables. See the description of the Variable Info command in the Options menu for additional information on setting the bounds.

The maximum number of times in which the equations are solved (i.e., the number of function calls) may be specified, along with the relative tolerance. Calculations will stop if: 1) the relative change in the independent variable(s) between two successive steps is less than the specified tolerance; or 2) the number of steps exceeds the specified maximum. EES will also stop the calculations if the equations cannot be solved with specified value(s) of the independent variables within the tolerance and allowable number of iterations specified with the Stopping Criteria command in the Options menu.

Min/Max Table provides the same capability as the Min/Max command, except that the calculations will be repeated for each row in the Parametric Table. (See the description of Parametric menu commands on the following pages for additional information on the use of the Parametric Table.) As with the Min/Max command, a dialog window will appear in which the variable to be maximized or minimized and the independent variable(s) can be selected. In this case, however, the variable which is to be optimized and all of the independent variables (whose values will be varied in seeking the optimum) must appear in the Parametric Table. The start and stop runs in the Parametric Table for which the calculations will be done may be specified. Values in the Parametric Table which are shown in normal type are fixed and are treated just as if they were set to that value with an equation in the Equations window. The variable which is to be optimized and the independent variable(s) must be the same for each run. If no errors are encountered, the optimum is computed and the values of the remaining columns in the table are entered for each run.

Uncertainty Propagation determines the uncertainty of a selected calculated variable as a function of the uncertainties of one or more measured values upon which it depends. In many cases, an important quantity is not directly measured by rather calculated as a function of one or more variables that are directly measured, i.e., Y = f(X1, X2, ...). The measured variables, X1, X2, etc. have with a random variability which is referred to as its uncertainty. In EES, that uncertainty is displayed with a ± symbol, e.g., X1 = 300 ± 2.

The purpose of this command is to calculate how the uncertainties in all of the measured variables propagate into the value of the calculated quantity, Y. The method for determining this uncertainty propagation is described in NIST Technical Note 1297 (Taylor B.N. and Kuyatt, C.E., Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results, National Institute of Standards and Technology Technical Note 1297, 1994). Assuming the individual measurements are uncorrelated and random, the uncertainty in the calculated quantity can be estimated as

$$\boldsymbol{U_Y} = \sqrt{\sum_i \left(\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X_i}} \right)^2 \boldsymbol{U_X^2}}$$

where U represents the uncertainty of the variable. After selecting this command, EES will present two lists of variables. Select the variable for which the uncertainty propagation is to be determined from the list on the left. Select one or more measured variables from the list on the right. Note that the variables appearing in the measured variables list must be constants so that their values are set to a numerical constant with an equation in the EES Equations window. To specify the uncertainty associated with the measured variables, click the Set Uncertainties button below the right list. A second dialog window will appear in which the absolute or relative (fraction of the measured value) uncertainties for each selected measured variable can be specified. An uncertainty value for each measured variable must be provided. Click the OK button to set the uncertainties and close the Specify Uncertainties dialog window. Click the OK button Uncertainty Propagation dialog to start the calculations.

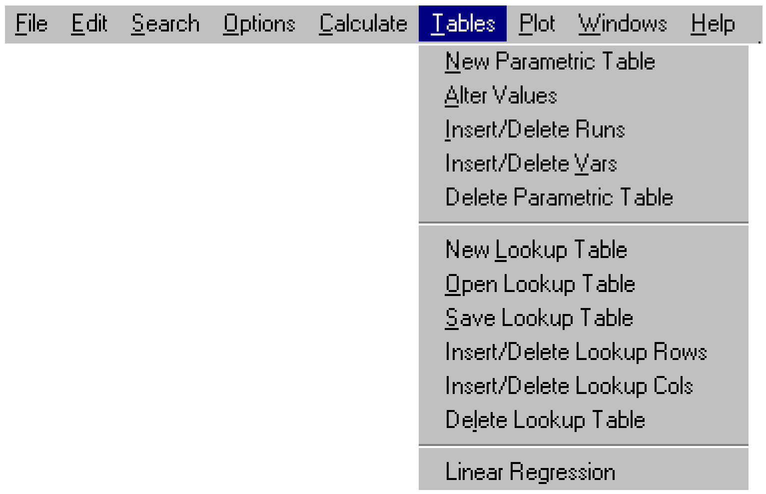
After the calculations are completed, EES will display an abbreviated Solution Window containing the calculated and measured variables and their respective uncertainties. The partial derivative of the calculated variable with respect to each measured variable will also be displayed.

Uncertainty Propagation Table provides the same function as the Uncertainty Propagation command, namely the determination of the uncertainty propagation in a calculated variable. The difference is that this command allows the uncertainty calculations to be repeated for one or more measurements by using the Parametric Table. The calculated and measured variables must all be in the Parametric Table before this command is used. After selecting the command, the Uncertainty Propagation dialog window will appear in which the calculated quantity is selected from list of variables in the left list and the measured variable(s) are selected from the list on the right. The Parametric Table calculations will proceed after the OK button is selected, just as if the Solve Table command were applied. The value and uncertainty for the calculated variable and each measured variable will be displayed in the Parametric Table after the calculations are completed. The calculated variable can then be plotted with error bars representing the propagated uncertainty using the New Plot Window command.

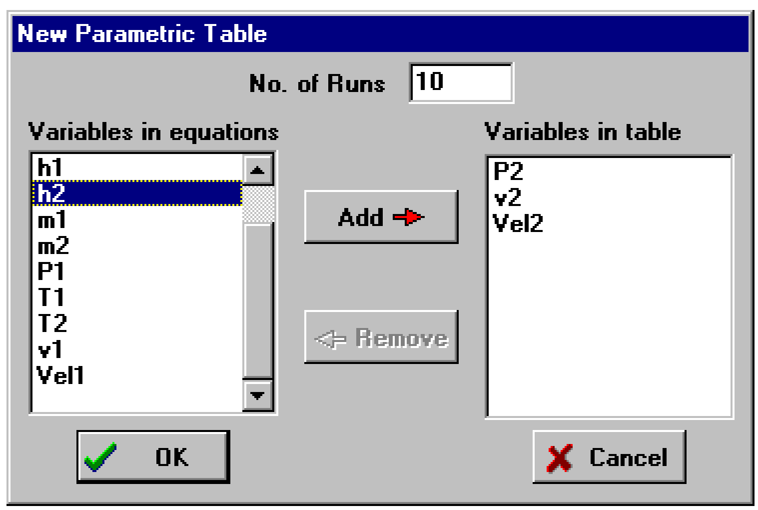
Update Guesses replaces the guess value of each variable in the Equations window with the value determined in the last calculation. This command is accessible after calculations have been successfully completed. Update Guesses improves the computational efficiency of an EES calculation since it ensures that a consistent set of guess values is available for the next calculation. Exactly the same function is provided with the Update button in the Variable Info dialog window, but the Update Guesses command is more accessible.

Reset Guesses replaces the guess value of each variable in the Equations window with the default guess value for that variable. Unless otherwise specified, EES assumes all guess values are 1.0. You can change the default guess values with the Default Info command in the Options menu. You should reset the guess values only if you are experiencing convergence difficulties and you have changed the guess values in an attempt to find a solution.

The Tables Menu



New Parametric Table creates a new Parametric Table after first deleting any existing Parametric Table. Parametric Tables are used in EES to automate repetitive calculations, to solve differential equations, and to present data for plotting or curve-fitting. A dialog window will appear in which information must be entered to create the table, as in this example.

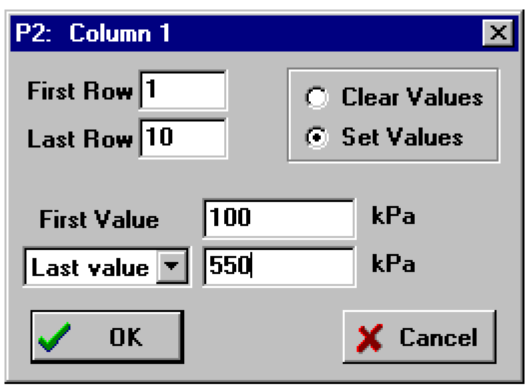


The number of runs, which corresponds to rows in the table, is entered in the field at the top. All variables (both independent and dependent) which are to appear in the table are selected from the alphabetical list of variables on the left. To select a variable, click on its name in the list, which will cause it to be highlighted. Multiple names can be selected. Click the Add button to move the highlighted variable names to the list on the right. (As a shortcut, a variable is automatically added to the list on the right when you double-click on its name in the list on the left.) The variables in the right-hand list will appear in the columns of the table in the same order in which they appear in the list. A variable can be removed from the table list by clicking on its name in the right list and then clicking the

Remove button or by double-clicking on the variable name. Pressing the OK button will create the Parametric Table, overwriting any existing table.

The Parametric Table operates somewhat like a spreadsheet. Numerical values can be entered in any of the cells. Entered values are assumed to be independent variables and are shown in normal type. Entering a value in a table produces the same effect as setting that variable to the value in the Equations window. Dependent variables will be determined and entered into the table in blue, bold, or italic (depending on the choice made in the Preferences dialog) when the Solve Table, Min/Max Table, or Uncertainty Propagation Table command is issued. If a variable is set in the table, it can not also be set in the Equations window; otherwise the problem will be overspecified. Each row of the table is a separate calculation. The independent variables may differ from one row to the next. However, for every row, the number of independent variables plus the number of equations must equal the total number of variables in the problem.

Alter Values provides an automatic way to enter or clear the values of a variable for multiple runs. There are two other ways of changing the data in the Parametric table. Clicking on the ▼ control at the upper right of each column header cell will bring up a dialog window which operates just like the Alter Values dialog shown below. Also, you can simply type the values directly into the Parametric Table.

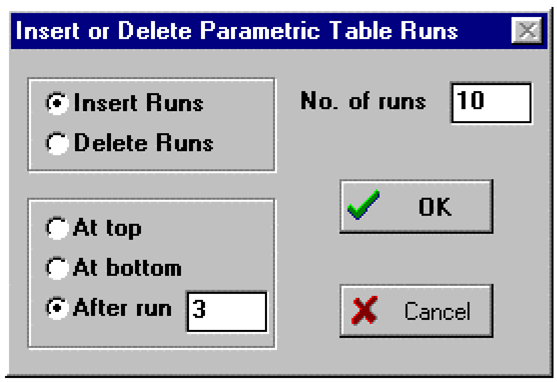


The runs (i.e., rows) affected are specified at the upper left of the dialog. The variable for which changes are to be made is selected from the list by clicking on its name. The column for this variable will be cleared if the Clear Values control is selected. If Set Values is selected, values for the selected variable will be entered automatically in the table starting with the value in the First Value field. The list box below the First Value controls the manner in which successive values in the table are generated. The choices are Last Value, Increment, and Multiplier. Increment or Multiplier result in successive values in the table being determined by either adding or multiplying, respectively, the preceding table value by the value provided in the box. If Last Value is selected (as shown) the increment will be selected such that the last run has the specified value. The Apply button will change the Parametric table as specified but control will remain in the

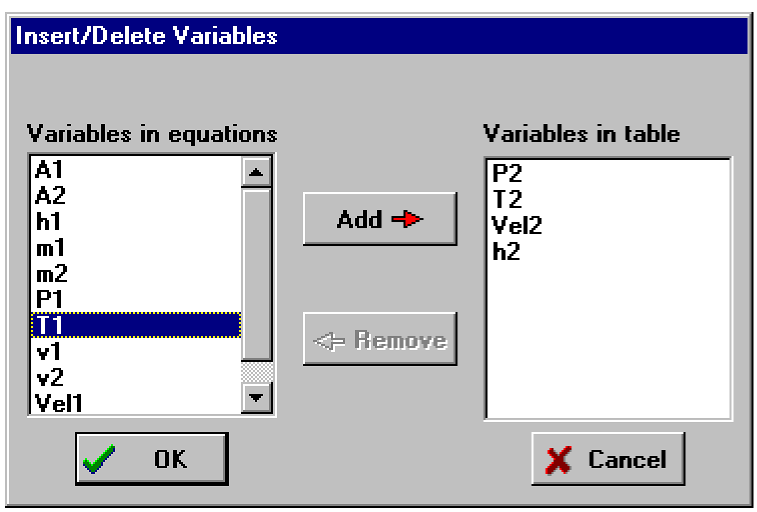
Alter Table Values dialog window so that additional changes can be made. The OK button accepts and finalizes all of the changes made to the Parametric table.

The numerical values entered in a table, either directly or through the Alter Values command, identify independent variables in the equation set; they are shown in normal type. Independent variables are fixed to a constant for each run, just as if there were an equation in the Equations window setting the variable to the constant. Dependent variables are shown in italic, blue or bold depending on the choice made in the Preferences dialog. These values are automatically entered in the table with the Solve Table and Min/Max Table commands. If a value is set in a table, it must not also be set in the Equations window; otherwise an error message will be displayed.

Insert/Delete Runs allows the number of runs in an existing Parametric Table to be changed by inserting or deleting one or more rows in the table at a specified position.



Insert/Delete Variables allows variables in an existing Parametric Table to be added or removed. The following dialog window will appear.



The list on the right shows the variables which currently appear in the Parametric Table. Variables which may be added to the table appear in the list on the left. To add one or more variables to the table, click on the variable name(s) causing them to be highlighted. Click the Add button to move the highlighted variable names to the list on the right. (You can also add a variable by double-clicking on the variable name.) Variables can be deleted from the table by selecting them from the list on the right, followed by clicking on the Remove button.

Variables will appear in columns of the Parametric Table in the same order as they appear in the list on the right. The order of the variables in this list can be changed by pressing and holding the left mouse button on a variable name while sliding it up or down to a new position in the list. The column order of an existing Parametric table can also be changed by clicking on the column header cell as described in Chapter 2.

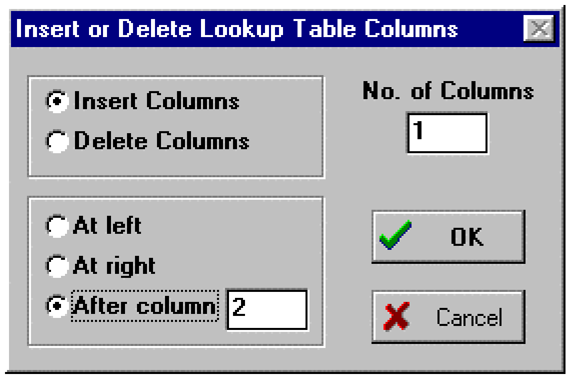
Delete Parametric Table will, after a confirmation, delete the Parametric Table and recover the memory it required.

New Lookup creates a table with the specified number of rows and columns in which tabular numerical data may be entered. If there is an existing Lookup Table, it will be overwritten when this command is completed. The tabular data may be automatically interpolated, differentiated, and used in the solution of the problem using the **Interpolate**, **Differentiate**, **Lookup**, **LookupRow**, and **LookupCol** functions described in Chapter 4. Only one Lookup Table may appear in the Lookup Table Window at any one time. However, the data in the Lookup Window may be saved in a Lookup File (with a .LKT filename extension) and the Lookup Files stored in disk files can also be accessed by the **Interpolate**, **Differentiate, Lookup**, **LookupRow**, and **LookupCol** functions. Lookup tables and files provide a great deal of power to EES by allowing any functional relationship between variables which can be represented by tabular information to be entered and used in the solution of the equations.

Open Lookup will read into the Lookup Table Window a Lookup file which was previously stored with the Save Lookup command or as a text file. A Lookup file is a two-dimensional table of data that has been stored in a disk file. A name and display format for each column of data may also be stored, depending on the file format. Lookup files can be accessed by the **Differentiate**, **Interpolate**, **Lookup**, **LookupCol**, and **LookupRow** commands. EES recognizes both binary and ASCII forms for Lookup files. Binary files are identified with a .LKT filename extension. ASCII Lookup files usually have a .TXT filename extension, although any filename extension (other than .LKT) can be used. Each file format has advantages and disadvantages. The binary form is read in more quickly and it requires smaller file sizes. The ASCII form is easier to edit and it can be written by spreadsheet or other applications. See Chapter 4, Using Lookup Files and the Lookup Table, for details.

Save Lookup copies the data in the Lookup Window into a Lookup file. Lookup files can be stored either as a binary file with the .LKT filename extension or as an ASCII file with a .TXT filename extension. The Lookup file can be later read with the Open Lookup Table command or used directly from the disk in the **Interpolate**, **Differentiate, Lookup**, **LookupRow**, and **LookupCol** functions. The information in the Lookup Table Window is also stored with the problem information when the Save or Save As commands are given. See Chapter 4, Using Lookup Files and the Lookup Table, for more information relating to Lookup tables and files.

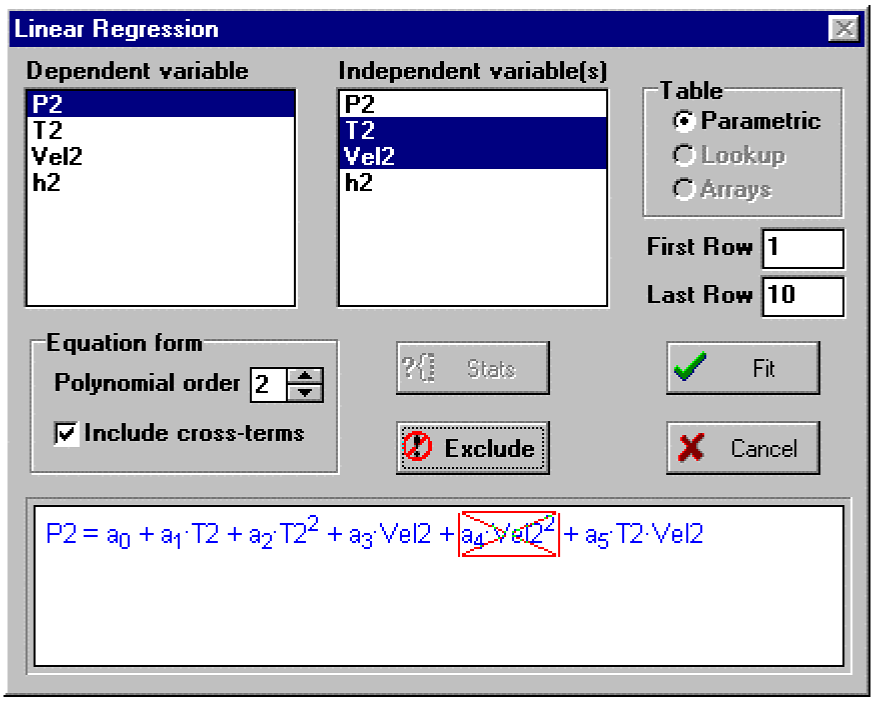
Insert/Delete Lookup Rows and Insert/Delete Lookup Cols allows one or more rows or columns to be inserted or deleted at a specified position in an existing Lookup Table.



Delete Lookup will delete the existing Lookup Table and recover the memory it required.

Linear Regression provides regression capability for the data in the Parametric, Lookup, or Arrays tables. Note that the Curve Fit command in the Plot menu also provides regression capability but only for one independent variable. With the Linear Regression command, the data in any column can be regressed as a function of the data in up to 6 other columns.

The dialog window shown below appears after the command is chosen. Select the table you wish to operate on from the radio buttons at the upper right and the starting and stopping rows in that table. Specify the dependent variable by clicking on the variable name in the list on the left. The independent variable(s) are selected by clicking on the names in the right list. To de-select an item, click it a second time.

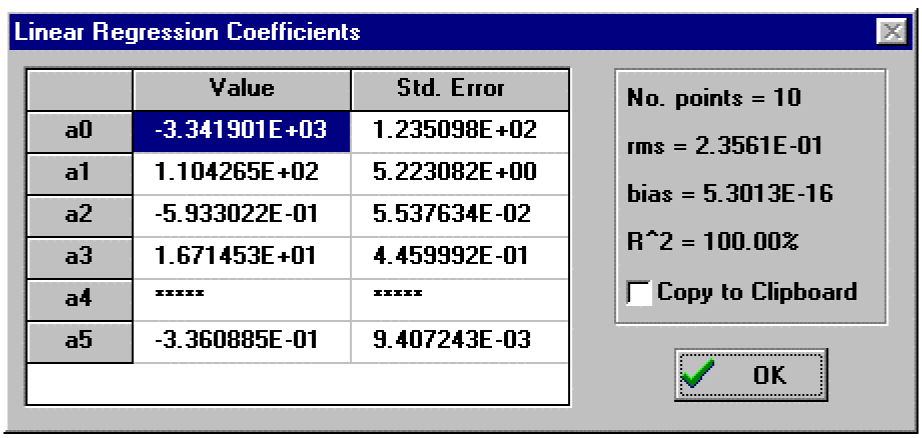


The dependent variable will be represented as a linear polynomial function of the independent variables. The order of the polynomial is set between 0 and 6 by clicking on the 'spin button' up or down arrows. If the cross-terms box is selected then terms involving the product of the independent variables will be included in the correlation. As any information relating to the equation form is entered, a representation of the equation to be fit is displayed in the box at the bottom as shown above.

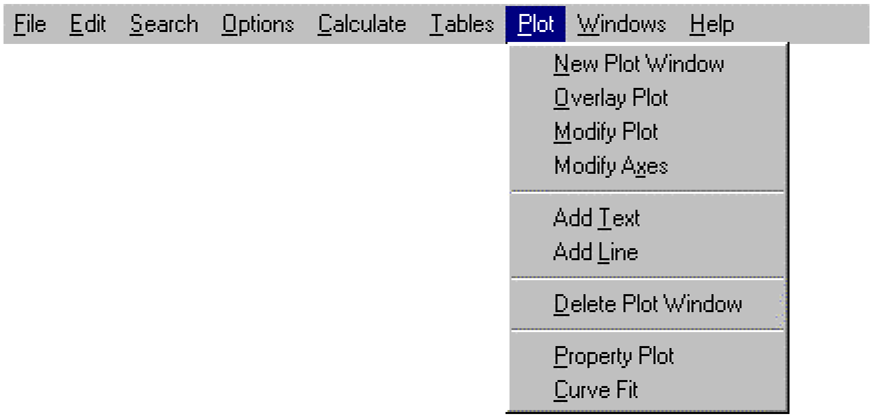
You may exclude some terms from the regression by clicking on the term. This action will display a box around the selected term and enable the Exclude button. Click the Exclude button to remove the term from further consideration. A removed term is displayed within a crossed-out red box as shown above. If you later wish to include an excluded term, click on it. The Exclude button will then be titled Include. Click the Include button.

When the form of the equation is that which you want to fit, click the Fit button. If the fitting process is successful, the fitted equation will appear in the display box. The Stats button will been enabled. Clicking the Stats button will provide a table listing all of the coefficients, their associated standard errors, and other statistics such as the root-mean-square (rms) error, the bias error, and the R^2 value, as shown below. Coefficients which have been excluded will be represented in the table with stars. The coefficients can be copied to the clipboard by checking the Copy to clipboard box.

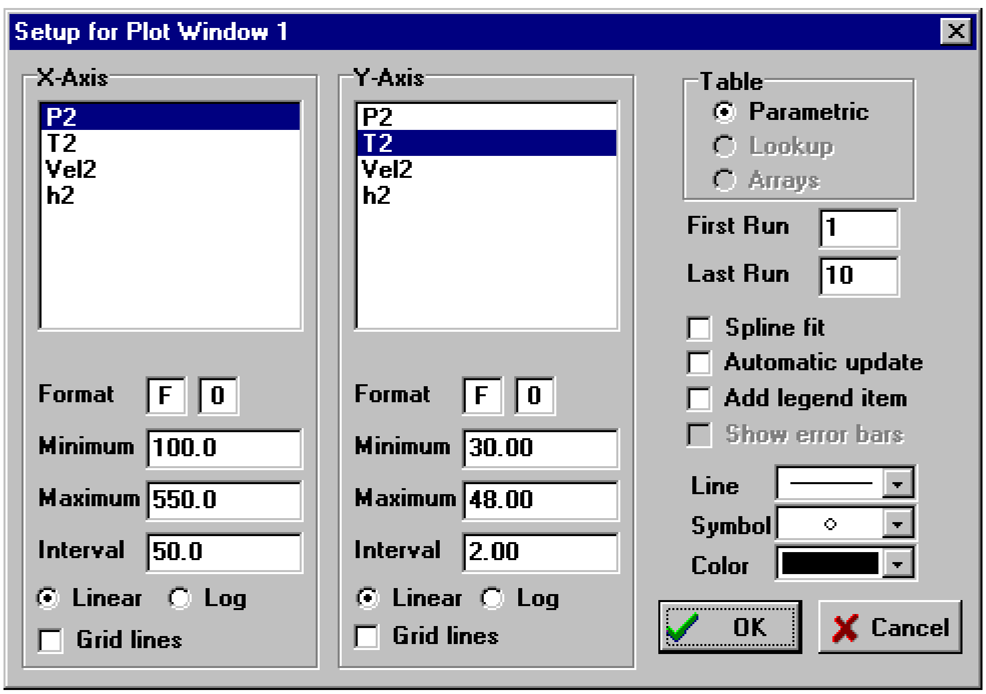
After a successful fitting process, the Fit button in the Linear Regression dialog window will be changed to Copy and the Cancel button will changed to Done. Either button will dismiss the dialog window. The Copy button will first copy the fitted equation to the clipboard. You can then paste this equation into the EES Equations window or into any other application that accepts text. Note, however, that the Copy process will overwrite any other information in the clipboard, such as the coefficients copied from the Linear Regression Coefficients dialog window.



The Plot Menu



New Plot Window allows any variable defined in the Parametric, Lookup, or Array Tables to be plotted as a function of any other variable in that table. There may be up to ten plot windows. Use the Overlay Plot command if you wish to plot in an existing plot window. The information needed to produce the plot is specified in the New Plot Window dialog. All of the information provided in this dialog window can later be changed using Modify Axes and Modify Plot commands and the Plot Window controls described in Chapter 2.



First, select the table from which you wish to plot using the radio button controls at the upper right of the window. Tables which are not defined will be shown in grayed type. The variables to be plotted on the x– and y–axes are selected by clicking on their names in the x– and y–Axis lists. EES will automatically select appropriate values for the number of display digits, the minimum and maximum axis values, and the interval when a variable is selected. All of these axis formatting variables may be changed.

The two fields to the right of the word Format contain pop-up menus which control the format of the numbers appearing in the scale for each axis. F and E format the numbers with a fixed number of decimal places or exponential notation, respectively. The number in the second field is the number of decimal places (for fixed notation) or significant figures (for exponential notation).

Grid lines will be drawn if the 'Grid Lines' checkbox control is selected. The number of grid lines and scale numbers is determined by the specified interval value.

The line type, symbol (or bar type), and color of the plot curve can be selected from their respective drop-down lists. The 'Spline fit' control will provide a spline-fit curve through the plotted points.

When the 'Automatic update' control is selected, the plot will be generated using the current data in the Parametric Table, rather than the data which existed when the plot was first drawn. The plot will be updated as the data in the table are changed.

If the 'Add legend item' is selected, a text item having the name of the Y-axis variable will be placed at the upper left corner of the plot, preceded by the line and symbol type used for the plot. The legend item text can be moved, changed, or deleted just as any plot window text item, as described in the Plot Window section of Chapter 2.

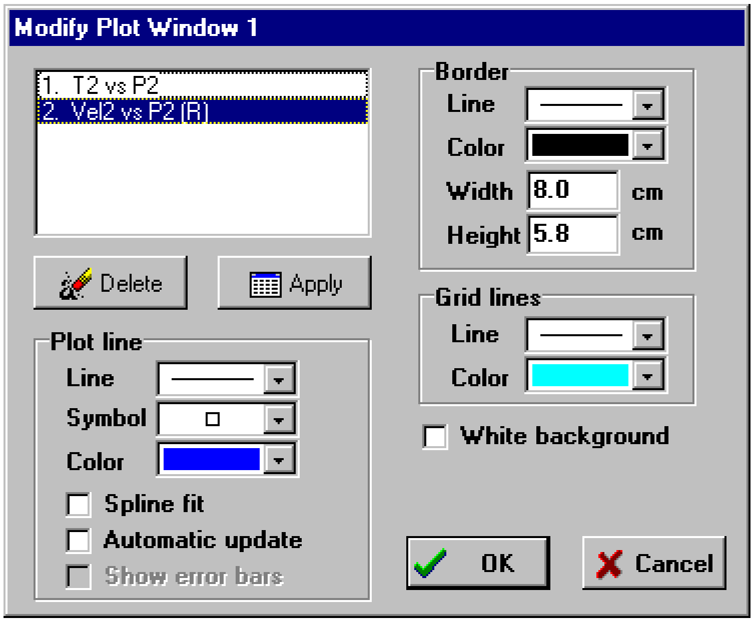
The "Show error bars" control is accessible only if one or both of the variables selected for plotting have associated uncertainty values specified with the Uncertainty Propagation Table commands.

Overlay Plot allows a new plot curve to be drawn over existing plots. The use of this command is identical to that for the New Plot command described above except that it does not first clear the Plot window. All overlaid plots should share the same x-axis scaling. If the scale specified for a plot overlay differs from that of the first plot, a control will be displayed to select the existing scale on the left-hand y-axis or a new scale on the right-hand y-axis. Following plot overlays may then use either the left or right y-axis scale, depending on the choice of the axis selection control.

Modify Plot allows the characteristics of existing plot curves to be changed by manipulation of information in the following dialog window. This command can also be invoked by double-clicking the mouse button within the plot rectangle.

The plot for which changes are to be made is selected from the list at the upper left. The plots appear in this list in the order in which they were generated. An (R) following the plot descriptor indicates that this plot uses the right-hand y-axis scale.

The line type, symbol, and color of the plot curve can be changed using the drop-down lists at the lower left. The 'Spline fit' and 'Automatic update' options may be changed. (See the New Plot Window command for a description of these options.)



Controls are provided to change the size and characteristics of the plot border and the grid lines. Click the Apply button after changing the plot line characteristics and before selected another plot. The plot window will immediately show any changes.

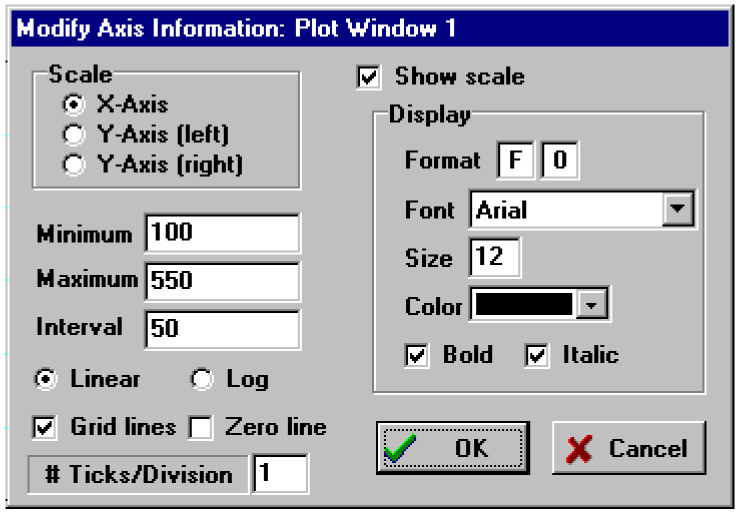
A single plot curve may be selectively deleted (leaving all other plots intact) using the Delete button. Legend text for the plot is also deleted. The Delete Plot Window command described below will delete an entire plot window, including all overlays.

Modify Axes allows the appearance of the axes of an existing plot to be changed. This command can also be invoked by double-clicking the mouse on the axis scale for which changes are to be made. The dialog window shown below will appear. The axis for which changes are to be made is selected with the radio-button controls at the upper left. The current minimum, maximum, and interval values for the selected axis are shown. These values can be changed and the plot will be redrawn scaled with the new values.

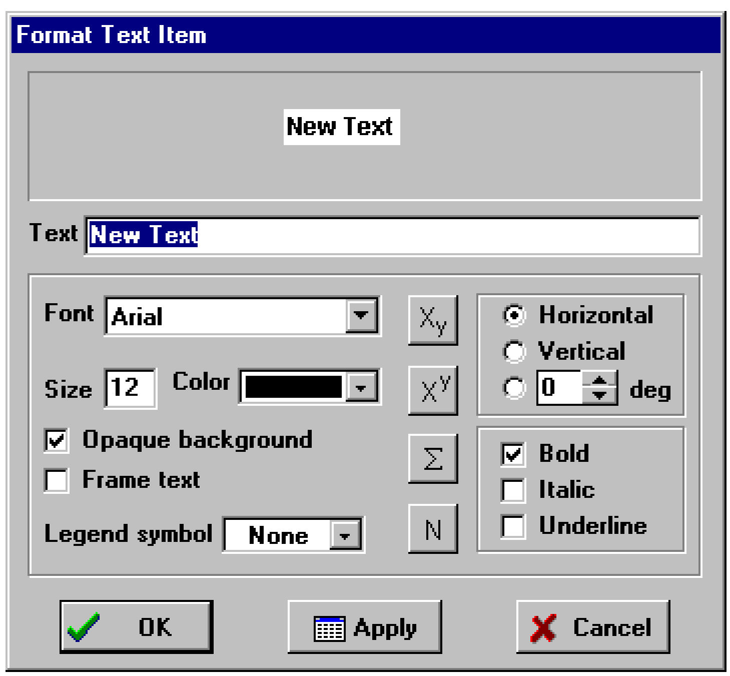
The No. Ticks/Division is the number of minor tick marks in each interval. If selected, Grid lines are normally placed at each major tick mark. However, the No. Ticks/Division toggles to No. Grids/Division if you click on this control. Grid lines can be placed at positions between the major ticks by setting the No. Grids/Division to a value greater than 0.

The display format, font, font size, font style, and color of the scale numbers can be changed using the drop-down menus appearing on the right of the dialog window. These fields will be hidden if the Show Scale checkbox is not selected, in which case a scale will not be drawn.

Clicking the axis control applies the changes so that they can be viewed in the Plot window. The OK button accepts the changes and exits the dialog.. The Cancel button will restore the plot to the condition it was in before this command was issued.



Add Text allows a line of text to be placed in the current plot window. A text item will be allocated with the string ‘New Text’ and with the characteristics of the last allocated text. The Format Text dialog shown below will then appear in which the text can be edited and formatted.



The font, font size, font style, and color of the text can be selected from the drop-down lists appearing below the text field. The orientation of the text may be horizontal or rotated 90° counterclockwise. The latter orientation is useful for labeling the y-axes. The Opaque control displays the text with a white background so that the background does not interfere with the text. Frame text places a border around the text.

The formatting capabilities for text are extensive and they are facilitated with the four speed buttons for superscript (X^y), subscript (X_y), symbol font (Σ), and normal font (N). To create a subscript, for example, select the text which is to be subscripted and then click the subscript button. The formatted text as it will appear in the plot window is shown at the top. Note that EES puts characters into the string to generate the subscript and other formatting options. The backslash (\ ) character is a special character used for formatting and it cannot be displayed in the text. You can enter a backslash to cancel a formatting option.

EES allows any horizontal text item to be associated with a legend symbol. The Legend drop-down list will display a description of each existing plot. If a plot is selected, the line type and symbol used for that plot will be displayed just to the left of the text item and it will move when the text item is moved. If the plot is deleted, the text item will also be deleted.

All text items can be moved by dragging them with the mouse as described in the Plot Windows section of Chapter 2. The characteristics of any text item, including the axis labels and plot title, can be altered by double-clicking the mouse within the text rectangle which will bring up the Format Text dialog window.

Add Line allows a line or arrow to be placed anywhere within the Plot window. After this command is issued, the cursor will appear as a cross. Press and hold the mouse button down at the position where you wish the line to begin. Hold the mouse button down while moving the mouse to the desired end position of the line and then release it.

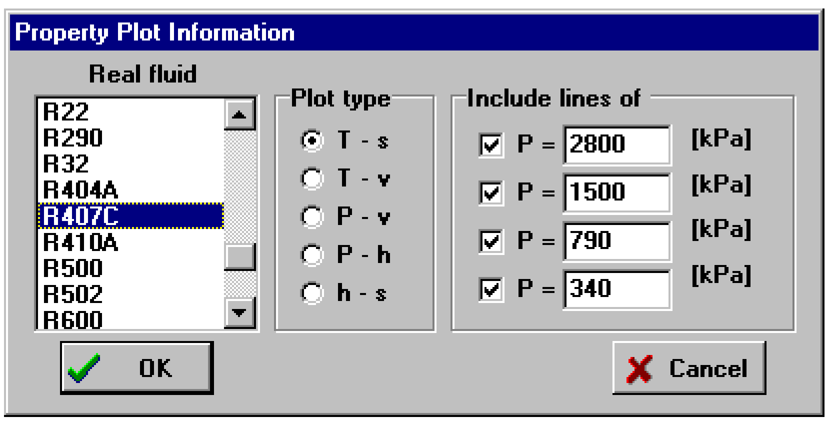
You can move, resize or rotate the line after it is created. To move the line, press and hold the mouse button down anywhere near the middle of the line while dragging it to its new position. To rotate the line and/or change its length, press and hold the mouse button on either end. Move the end to its new position and release the mouse button. Initially, a line will be created with the default characteristics of the previous line or with default characteristics. If you wish to change the characteristics, double-click on the line. A dialog window will appear in which the line and arrowhead characteristics may be selected.

Delete Plot Window will delete the entire contents of the selected Plot Window. Use the Delete button in the **Modify Plot** dialog window if you wish to delete only one of several overlayed plots.

Property Plot creates a new plot window with thermodynamic property data for a selected substance. Once created, additional property data or thermodynamic cycle state points can be superimposed on the plot using the **Overlay Plot** command. Also, the plot characteristics and axis scales can be modified in the usual manner with the **Modify Axes** and **Modify Plot** commands.

Select the substance from the list at the left. The substance type (real fluid or ideal gas) is shown above the list. The general rule is that the substance is modeled as a real fluid if its name is spelled out (e.g., Oxygen) and as an ideal gas if its name is a chemical formula (e.g., O2). Air is the exception to this rule.

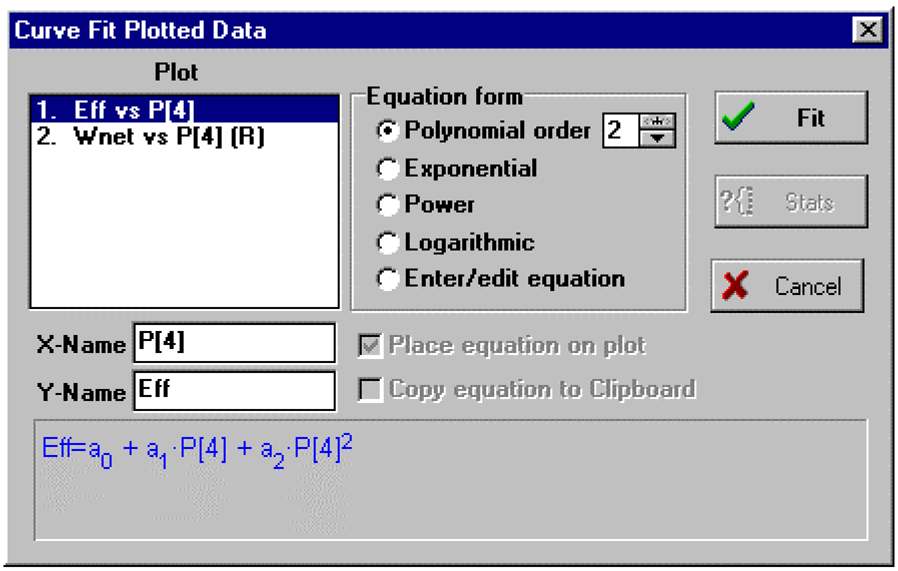
For all substances except AIRH2O (psychrometric air-water mixtures), there are four buttons which allow a specification of Temperature-entropy (T-s), Temperature-volume (T-v), Pressure-volume (P-v) or Pressure-enthalpy (P-h) coordinate systems. The AIRH2O substance provides a field in which the total pressure can be specified.



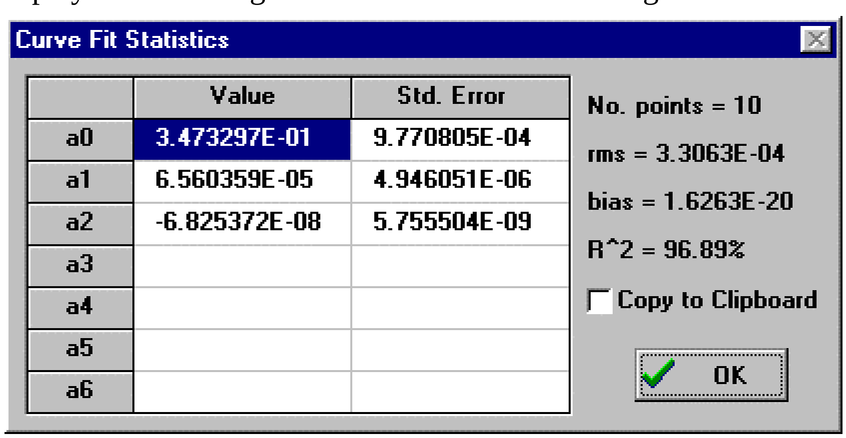
At the right are controls which allow specification of four isobars or isotherms. A line of constant pressure or temperature with the value shown in the box will be superimposed on the plot. Suggested values are provided. If you do not wish to display the isobar or isotherm, click on the check box preceding the value.

Curve Fit will find the best fit of a smooth curve through a previously plotted set of data points using unweighted least squares. (The Curve fit dialog provides a fit with a single independent variable. The Linear Regression command in the table menu allows a variable to be fitted with as many as 6 independent variables.) The dialog window shown below will appear.

Chose the data to be fitted from the list of plots at the left. Note that data can be plotted from the Parametric Table, the Lookup Table, or the Arrays Table with the New Plot or Overlay Plot commands. Select the form of the curve fit by clicking the appropriate radio button. A sample of the equation form will appear in blue in the box at the bottom of the dialog window. The first four buttons correspond to commonly used equation forms for which linear least squares can be used to determine the unknown coefficients. The Enter/edit equation button allows you to enter any equation form or to edit a previously entered equation. The equation you enter may be linear or non-linear in the unknown parameters. You will prompted to supply guess values bounds for the unknown parameters.



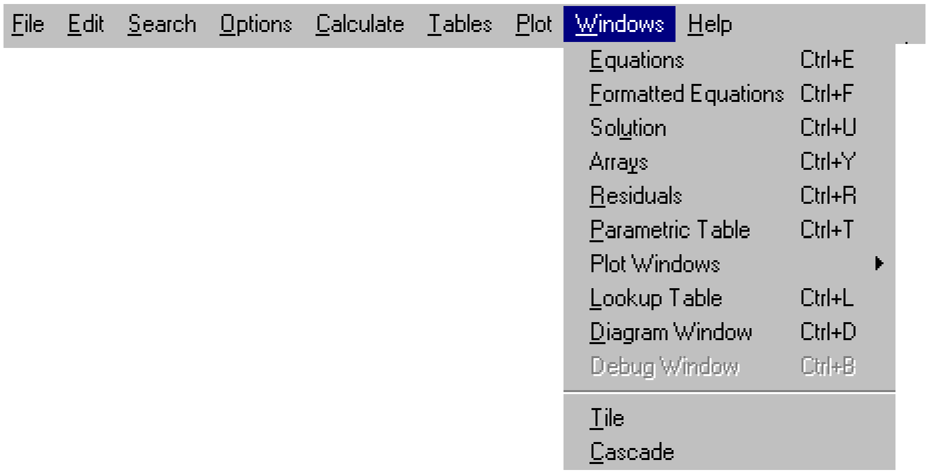
Click the Fit button (or press the Enter key). The fitted equation will be displayed in the box at the bottom of the dialog window. A Stats button will appear. Clicking the Stats button will display the following statistical information relating to the curve fit.



Std. Error is the standard error of the curve-fitted parameter value; rms is the root mean square error of the fit; bias is the bias error of the fit. R^2 is the ratio of the sum of squares due to regression to the sum of square about the mean of the data.

The Fit button will now have changed into the Plot button. Click the Plot button if you wish to have the curve fit equation overlayed on your plot. If the Plot Legend check box is selected, a legend containing the equation will be created and displayed on the plot. The curve fit equation will be copied to the clipboard if the To Clipboard checkbox is selected when either the Plot or Cancel button is selected.

The Windows Menu



Equations causes the Equations window to become the active window by bringing it to the front of all other windows and making it visible if it were hidden previously.

Formatted Equations first checks the syntax of the equations and then brings the Formatted Equations window to the front displaying the contents of the Equations window in mathematical format.

Solution, Arrays and Residuals cause the Solution, Arrays, and Residual windows, respectively, to be moved to the front of all other windows. These windows are normally viewed after the Solve, Min/Max, or Uncertainty Propagation command has been successfully completed. Any change made to the Equations window will remove these windows from the screen if the Hide Solution after Change Option in the Preferences dialog (Options tab) is selected. If EES is unable to solve the equation set and terminates with an error, the name of the Solution window will be changed to Last Iteration Values and the values of the variables at the last iteration will be displayed in the Solution window; the residuals for the last iteration will be displayed in the Residuals window.

Plot Windows will display a submenu with items for each of the 10 possible plot windows. Selecting a plot window will bring that window to the front of all other windows. The Plot menu commands, e.g., Modify Plot and Modify Axes, will operate without asking for a plot window designation when one of the plot windows is foremost. The menu item will be grayed if the corresponding plot window is not defined. The graphics in any of the plot windows can be copied to the Clipboard by selecting Copy from the Edit menu.

Parametric Table and Lookup Table bring the Parametric and Lookup Table windows, respectively, to the front of all other windows and make it the active window. The

Parametric and Lookup Table windows may be hidden by choosing close from the Windows control menu or by pressing Ctrl-F4.

Diagram will bring the Diagram to the front of all other windows. A diagram is entered into EES from a drawing program. Copy the diagram in the drawing program, switch to EES and bring the Diagram window to the front. Then use the Paste command to move the diagram into the window. Note that values can be entered from the Diagram window only if it is currently displayed on the screen.

Debug Window will be generally be disabled as shown. This window is available only after the Solve command is attempted with the number of equations not equal to the number of variables. In this case, an option is presented to display the Debug Window and this menu item becomes enabled. Any change made to the Equations window will clear the Debug window and disable this menu item.

Tile arranges all open windows to fill the screen so that a portion of each can be viewed.

Cascade arranges the currently visible windows so that the title bar of each is shown.

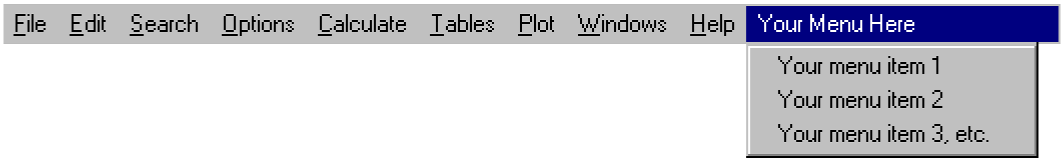
The Help Menu

Help Index will activate the Help processor which provides specific information on the use of EES. The Help processor will open to the EES Information index which lists the subjects for which help is available. Clicking on the subject opens the Help window to information for that subject. Help can also be accessed by pressing the F1 key which will bring up help information specific to the window or dialog which is foremost. The on-line help provides most of the information contained in this manual.

Using Help shows information provided by the Windows Help processor on how to use the features in the Help program.

About EES will bring up the EES header window. This window registration information and indicates the version of your EES program and the amount of available memory. This information will be needed in any correspondence with F-Chart Software.

The Textbook Menu



The Textbook menu is perhaps misnamed. It is really a user-defined menu that is designed to allow easy access to EES files. However, it has been used to provide a convenient means to access EES problems associated with a textbook, and thus its name. This menu can be created either by opening a textbook index file with the Load Textbook command (File Menu) or by placing the textbook index file in the USERLIB subdirectory.

A textbook index file is an ASCII file identified with the filename extension (.TXB). When EES reads a textbook index file, it creates the Textbook menu at the far right of the menu bar, as shown above. The format of the textbook index file is quite simple. The menu shown above was created with the following textbook index file.

```
Your Menu Here
1
Textbook information line 1
Textbook information line 2
Textbook information line 3
Reserved
>Your menu item 1
Descriptive problem name1 | FileName1.EES | HelpFile1.HLP | NO.BMP
Descriptive problem name2 | FileName2.EES | HelpFile2.HLP | NO.BMP
Descriptive problem name3 | FileName3.EES | HelpFile3.HLP | NO.BMP

>Your menu item 2
Descriptive problem name1 | FileName1.EES | HelpFile1.HLP | NO.BMP
Descriptive problem name2 | FileName2.EES | HelpFile2.HLP | NO.BMP
Descriptive problem name3 | FileName3.EES | NO.HLP | NO.BMP
>Your menu item 3, etc.
etc | FileName1.EES | HelpFile1.HLP | NO.BMP
```

The first line in the file is the menu title. This title is the name of the menu which will appear in the menu bar to the right of the Help menu. The following line is a version number used internally by EES. EES currently ignores this line but a 1 should be provided, as shown. The following three lines provide information about the textbook or problem set. This information will be displayed whenever any menu item is selected from the Textbook menu. A fourth line containing the word "reserved" is provided for possible future use. EES currently ignores the information on this line, but it must be provided. The following lines contain a menu item name (preceded by an identifying > character) and then one or more problem descriptions. The menu item name is the name that will appear in the Textbook menu list.

Each problem description line contains four pieces of information, separated by a | character. The first item is a descriptive name for the problem, which may be up to 128 characters. The second item is the filename for the EES program file. This filename may be partially or fully qualified with directory information, e.g., C:\myBook\Chapter1\Problem1.EES. However, in most cases, directory information is not necessary and it should not be included. The third item is an optional help file which is to be associated with this file. The help file can be an ASCII text file or a Windows HLP file produced by a Help formatting program. By convention, the help files have a .HLP filename extension. However, EES will accept any filename extension for either type of help file. If no help file is available, enter NO.HLP for this field. The final item is the filename for a figure associated with the problem. EES does not currently use the figure, but a figure name must be provided. Use NO.BMP as a placeholder.

The textbook index file should be placed in the same location (i.e., floppy disk, subdirectory, or folder) with all of the referenced EES program and help files. When the user selects a command from the Textbook menu, a dialog window will appear showing a list of the descriptive names of the problems for that menu item. The user can then select a name from the list and the file associated with that problem will be opened.

CHAPTER 4

Built-in Functions

EES has a large library of built-in mathematical functions. Many of these (e.g., Bessel, hyperbolic, error functions, etc.) are particularly useful for engineering applications. EES also provides a function to convert units and functions that help manipulate complex numbers. The major feature that distinguishes EES from other equation solving programs, however, is its extensive library of built-in functions for thermophysical properties. Thermodynamic and transport properties of steam, R22, R134a, R407C, air, ammonia, carbon dioxide, and many others are implemented in a manner such that any independent set can be used to determine the remaining unknown properties. EES also provides a Lookup Table which allows tabular data to be entered and used in the solution of the equations set. The first two sections of this chapter provide reference information for the built–in mathematical and thermophysical functions. The third section provides information on the use of the Lookup Table. Much of the information provided in this chapter can also be obtained from within the program using the Info button in Function Info dialog.

Mathematical Functions

The mathematical functions built into EES are listed below in alphabetical order. (The functions which operate on the Lookup Table are described in the Using the Lookup Table section at the end of this chapter.) All of the functions (except **pi** and **tableRun#**) require one or more arguments which must be enclosed in parentheses and separated with commas. The argument may be a numerical value, a variable name, or an algebraic expression involving values and variables.

abs(X**)** returns the absolute value of the argument. In complex mode, **abs** returns the magnitude of the complex argument. (See also **Magnitude**(X));

angle(X**)**, **angleDeg(**X**)** and **angleRad(**X**)** all return the angle (also called amplitude or argument) of complex variable X. Representing X as X_r + i*X_i, this function returns arctan(X_i/X_r). Angle will return the angle in either degrees or radians depending on the Trig Function setting in the Unit System dialog. AngleDeg will always return the angle in degrees and AngleRad will always return the angle in radians. All three functions return the angle in the correct quadrant of the complex plane. Note that the Angle functions are used to extract the angle of a complex number variable or expression but they cannot be used for assigning the angle of a complex number. For example, the equation Angle(X)=4 will produce an error.

arcCos(X**)** returns the angle which has a cosine equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcCosh(X**)** returns the value which has a hyperbolic cosine equal to the value of the argument.

arcSin(X**)** returns the angle which has a sine equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcSinh(X**)** returns the value which has a hyperbolic sine equal to the value of the argument.

arcTan(X**)** returns the angle which has a tangent equal to the value of the argument. The units of the angle (degrees or radians) will depend on the unit choice made for trigonometric functions with the Unit System command.

arcTanh(X**)** returns the value which has a hyperbolic tangent equal to the value of the argument.

bessel_I0(X**)** returns the value of zeroth-order Modified Bessel function of the first kind for argument value X where -3.75≤ X <infinity.

bessel_I1(X**)** returns the value of first-order Modified Bessel function of the first kind for argument value X where -3.75≤ X<infinity.

bessel_J0(X**)** returns the value of zeroth-order Bessel function of the first kind for argument value X where -3≤ X <infinity.

bessel_J1(X**)** returns the value of first-order Bessel function of the first kind for argument value X where -3≤ X <infinity.

bessel_K0(X**)** returns the value of zeroth-order Modified Bessel function of the second kind for argument value X where 0≤ X <infinity.

bessel_K1(X**)** returns the value of first-order Modified Bessel function of the second kind for argument value X where 0≤ X <infinity.

bessel_Y0(X**)** returns the value of zeroth-order Bessel function of the second kind for argument value X where 0< X <infinity.

bessel_Y1(X**)** returns the value of first-order Bessel function of the second kind for argument value X where 0< X <infinity.

cis(X**)** is a complex mode function that returns cos(X)+i*sin(X). The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command. However, you can append deg or rad to the angle to override the Unit System setting. For example, V=3*cis(20deg) will set the value of complex variable V to have a magnitude of 3 and an angle of 20 degrees, regardless of the Unit System setting.

conj(X**)** returns the complex conjugate of a complex variable X. Representing X as X_r + i*X_i, this function returns X_r - i*X_i. Note the this function returns a complex result. The EES equation, Y=conj(X),will set the real part of Y (Y_r) to the real part of X and the imaginary part of Y to negative of the imaginary part (Y_i).

convert('From', 'To') returns the conversion factor needed to convert units from the unit designation specified in the 'From' string to that specified in the 'To' string. The single quote marks are optional. For example, FI = **convert**(ft^2, in^2) will set FI to a value of 144 because 1 square foot is 144 square inches. Combination of units and multiple unit terms may be entered. In a combination of units, such as Btu/hr-ft^2-R, the individual units are separated with dash (i.e., minus) or division signs. Only one division symbol may be used in any one term. All units to the right of the division symbol are assumed to be in the denominator (i.e., raised to a negative power.) The ^ symbol is optional so ft2 and ft^2 are equivalent. The **convert** function will accept multiple unit terms if each term is enclosed within parentheses. Terms are separated with an optional * symbol or with a / symbol, as in the example below.

P =15* **Convert**((lbm/ft3)*(ft)/(s^2/ft), kPa)

The defined unit symbols can be displayed with the Unit Conversion Info command in the Options menu. If you find that a unit you need is not defined, you can enter it by editing the UNITS.TXT file in the EES directory.

cos(X**)** will return the cosine of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

cosh(X**)** will return the hyperbolic cosine of the value provided as the argument.

differentiate('Filename', 'ColName1', 'ColName2', ColName2=Value**)** returns the derivative determined from two columns of tabular data based on cubic interpolation. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

erf(X**)** returns the Gaussian Error function of X.

erfc(X**)** returns the complement of the Gaussian Error function of X which is 1-erf(X).

exp(X**)** will return the value e raised to the power of the argument X.

if (A, B, X, Y, Z**)** allows conditional assignment statements. If A<B; the function will return a value equal to the value supplied for X; if A=B, the function will return the value of Y; if A>B, the function will return the value of Z. In some problems, use of the **if** function may cause numerical oscillation. It is preferable to use the *if then else*, *repeat until* and *goto* statements in a function or procedure for conditional assignments. See Chapter 5 for additional information.

imag(X**)** returns the imaginary part of a complex variable X. Representing X as X_r + i*X_i, this function returns X_i. Note that the Imag function is used to extract the imaginary part of a complex number variable or expression. It cannot be used for assigning the imaginary part of a complex number. For example, the equation Imag(X)=4 will produce an error. Instead you should just enter X=4*i which will set X to 0 + 4*i. If you wish to only set the imaginary part of X, you can enter X_i=4.0

integral(Integrand, VarName**)** or **integral**(Integrand, VarName, Start, Stop, Step) returns the integral of the expression represented by Integrand with respect to the variable VarName, i.e., ∫(Integrand) d(VarName). There are two basic forms of the integral function which differ in their reliance on the Parametric table. If the values of Start, Stop, and Step are not provided, the **integral** function is used only in conjunction with the Parametric Table. In this case, VarName must be a legal variable name which has values defined in one of the columns of the Parametric table and Integrand can be a variable or any algebraic expression involving VarName and other variables or values. If Start, Stop, and (optionally) Step are provided, EES will numerically integrate all equations involving variable VarName, setting the value of VarName to values between Start and Stop as appropriate. If Step is not provided, EES will internally chose a step size using an automatic stepsize adjustment algorithm. The **integral** function can be used to solve initial value differential equations. See Chapter 7 for additional information.

interpolate('Filename', 'ColName1', 'ColName2', ColName2=Value**)** returns an interpolated or extrapolated value from tabular data in the Lookup table, a Lookup file (if filename is provided), or the Parametric table using cubic interpolation. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

interpolate1('Filename', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses linear interpolation.

interpolate2('Filename', 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses quadratic interpolation.

lookup(*'Filename'*, Row, Column**)** returns the value in the Lookup Table or Lookup file at the specified row and column. Filename is optional. See ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

lookupCol(*'Filename'*, Row, Value**)** uses the data in the specified row of the Lookup Table or Lookup file to determine the column which corresponds to the value supplied as the second argument. Filename is optional. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

lookupCol(*'Filename'*, Row, Value**)** uses the data in the specified row of the Lookup Table or Lookup file to determine the column which corresponds to the value supplied as the second argument. Filename is optional. See the ***Using Lookup files and the Lookup Table*** section of this chapter for more information and examples.

ln(X**)** will return the natural logarithm of the argument.

log10(X**)** will return the base 10 logarithm of the argument.

magnitude(X**)** returns the magnitude (also called modulus or absolute value) of a complex variable X. In complex mode, the **abs** function also returns the magnitude. Representing X as X_r + i*X_i, this function returns sqrt(X_r^2+X_i^2). Note that the **magnitude** function is used to extract the magnitude of a complex number variable or expression. It cannot be used for assigning the magnitude of a complex number. For example, the equation magnitude(X)=4 will produce an error.

max(X1, X2, X3, …**)** will return the value of the largest of its arguments. The number of arguments must be greater or equal to 1.

min(X1, X2, X3, …**)** will return the value of the smallest of its arguments. The number of arguments must be greater or equal to 1.

pi is a reserved variable name which has the value of 3.1415927.

product(Arg, Series_info**)** returns the product of a series of terms. Arg can be any algebraic expression. Series_info provides the name of the product index variable and the lower and upper limits which must be integers or variables which have been previously set to integer values. **product** (j, j=1,4) will return 1*2*3*4 or 24, which is 4 factorial. The **product** function is most useful when used with array variables, e.g., X[j]. For example, the product of the square of all 10 elements in the vector X can be obtained as **product** (X[j]*X[j], j=1,10).

real(X**)** returns the real part of a complex variable X. Representing X as X_r + i*X_i, this function returns X_r. Note that the **real** function is used to extract the real part of a complex number variable or expression. It cannot be used for assigning the real part of a complex number. For example, the equation real(X)=4 will produce an error. Instead you should just enter X=4 which will set X to 4 + i*0. If you wish to only set the real part of variable X, you can enter X_r=4.

round(X**)** will return a value equal to the nearest integer value of the argument.

sin(X**)** will return the sine of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

sinh(X**)** will return the hyperbolic sine of the value provided as the argument.

sqrt(X**)** will return the square root of the value provided as the argument which must be greater than or equal to zero.

step(X**)** will return a value of 1 if the argument is greater than or equal to zero; otherwise the Step function will return zero. The **step** function can be used to provide conditional assignments, similar to the if function. The **step** and if functions are provided to maintain compatibility with earlier versions. Conditional assignments are more easily and clearly implemented with the IF THEN ELSE statement in functions or procedures as described in Chapter 5.

sum(Arg, Series_info**)** returns the sum of a series of terms, i.e., ΣArg. Arg can be any algebraic expression. Series_info provides the name of the summation index variable and the lower and upper limits. These limits must be integers or variables which have been previously set to integer values. The function is best explained by examples. **sum**(j, j=1,4) will return 1+2+3+4 or 10. The **sum** function is most useful when used with array variables, e.g., X[j]. For example, the scalar product of two vectors, X and Y, each with 10 elements can be obtained as **sum**(X[j]*Y[j], j=1,10). See Chapter 7 for information on how the **sum** function can be used with array variables to manipulate matrices.

tableRun# returns the Parametric Table run number, i.e., the current row in the Parametric Table or zero, if the Parametric Table is not being used in the calculations. This function should only be used with the Solve Table or Min/Max Table command in the Calculate menu.

tableValue(Row, Column**)** or **tableValue(**Row, ‘VariableName’**)** returns the value stored in a specified row and column of the Parametric Table. The column number may be either entered directly as an integer number or indirectly by supplying the variable name for the desired column, enclosed by the single quotes, e.g., TableValue(6,’ABC’)[7]. An error message will be generated if the row or column (or corresponding variable name) does not exist in the Parametric Table or if the referenced cell does not have a value. The **tableValue** function is useful in the solution of some 'marching-solution' type problems in which the current value of a variable depends on its value in previous calculations.

tan(X**)** will return the tangent of the angle provided as the argument. The required units (degrees or radians) of the angle is controlled by the unit choice made for trigonometric functions with the Unit System command.

tanh(X**)** will return the hyperbolic tangent of the value provided as the argument.

trunc(X**)** will return a value equal to the integer value corresponding to the argument rounded toward zero.

UnitSystem(‘Unittype’**)** is a function which allows an EES program to know what unit settings have been selected with the Unit System command. This function takes one argument which must be placed within single quote marks. Legal arguments are 'SI', 'Eng', 'Mass', 'Molar', 'Deg', 'Rad', 'kPa', 'bar', 'psia', 'atm', 'C', 'K', 'F', and 'R'. The function returns either 1 (for true) or 0 (for false). As an example, the following assignment statement in an EES function or procedure,

```
g:=unitsystem('SI') + 32.2*unitsystem('Eng')
```

will set g equal to 1 if the user has selected the SI unit system and g equal to 32.2 if the user has selected the English unit system.

---

7 For compatibility with earlier versions, EES will also accept the #symbol preceding the variable name in place of enclosing it within single quotes, e.g., TableValue(6, #ABC).

Thermophysical Property Functions

The first argument of all built-in thermophysical property functions is the name of the substance. The substance names recognized by EES are[8]:

Substance Names for Thermophysical Property Functions

Air	*CH2*	*Oxygen*	*R123*
AirH2O	*H2O*	*Propane*	*R134a*
Ammonia	*Helium*	*R11*	*R141b*
C2H6	*Methane*	*R12*	*R500*
C3H8	*n-Butane*	*R13*	*R502*
C4H10	*N2*	*R14*	*R600a*
CarbonDioxide	*Nitrogen*	*R22*	*SO2*
CH4	*NO*	*R32*	*Steam (Water)*
CO	*NO2*	*R113*	*Steam_NBS*
CO2	*O2*	*R114*	

Additional substances that are provided with the 32-bit version of EES in external files in the USERLIB subdirectory are R113, R125, R143a, R404A, R407C, R410A, and R600a. These substances are provided with a user-supplied property file having a .MHE filename extension, as described in Appendix C. Also provided in the USERLIB subdirectory are the external routines providing thermodynamic property data for lithium bromide-water mixtures (H_LIBR, T_LIBR, V_LIBR, Q_LIBR, P_LIBR, X_LIBR), ammonia-water mixtures (NH3H2O) and specific heat, enthalpy and entropy for hundreds of additional substances with JANAF table references (JANAF). Documentation for these routines is provided through the Function Info command in the Options menu. Click the External routines button at the top right and then select the external routine name from the list. Clicking the Info button at the bottom of the dialog will provide the documentation.

It may appear from the above list that some substances, e.g., *N2* and *Nitrogen*, *CO2* and *CarbonDioxide*, *H2O* and *Steam* (or *Water*), are duplicated, but this is not true. Whenever a chemical symbol notation (e.g., *N2, CO2, CH4* etc.) is used, the substance is modeled as an ideal gas and the enthalpy and entropy values are based on JANAF table references. The JANAF table reference for enthalpy is based on the elements having an enthalpy value of 0 at 298K (537R). The entropy of these substances is based on the Third Law of Thermodynamics. Whenever the substance name is spelled out (e.g., *Steam* (or *Water), Nitrogen, R12, CarbonDioxide*, *Methane*, etc.) the substance is modeled as a real fluid with subcooled, saturated, and superheated phases. Exceptions to this rule occur for *Air* and *AirH2O,* both of

[8] Your version of EES may have additional fluids. Property data may be added as described in Appendix D.

which are modeled as ideal gases. *AirH2O* is the notation for air-water vapor mixtures, i.e., psychrometrics. Property information for up to 150 additional fluids can be added by the user as explained in Appendix C.

The property keywords *Water* and *Steam* are treated identically. Either keyword provides access to approximate water property functions based on empirical correlations which have been developed for rapid calculations. The *Steam/Water* property correlations assume the fluid is incompressible in the subcooled region; This assumption is not accurate for pressures above 350 atm and for states near the critical point. The *Steam_NBS* keyword uses property correlations published by Harr, Gallagher, and Kell (Hemisphere, 1984). These property correlations are extremely accurate at any condition. However, they require considerably more computing effort than the Steam/Water relations.

Many of the thermodynamic functions can take alternate sets of arguments. For example, the **enthalpy** function for steam can be accessed with temperature and pressure as arguments; alternatively, the same function could be accessed with entropy and quality as arguments. In general, any independent set of arguments can be supplied for thermodynamic functions. The transport functions (**conductivity** and **viscosity**), however, require temperature as the argument (for ideal gas substances) or temperature and pressure (for real fluids).

All arguments in thermophysical property functions, aside from the substance name, are identified by a single case-insensitive letter followed by an equal sign. The value or algebraic expression representing the value of the argument follows the equal sign. The letters which are recognized in function arguments and their meaning are as follows:

Property Indicators for Use in Thermophysical Functions	
B = Wetbulb Temperature	**T** = Temperature
D = Dewpoint Temperature	**U** = Specific Internal Energy
H = Specific Enthalpy	**V** = Specific Volume
P = Pressure	**W** = Humidity Ratio
R = Relative Humidity	**X** = Quality
S = Specific Entropy	

Arguments must be separated with commas and may be in any order, provided that the substance name is first, as in the examples shown below. EES will display the function name in the format selected for Functions in the Display Options dialog window. The substance name is an EES keyword and it will be displayed in the format selected for Keywords in the Display Options dialog window

EES does not require the argument to a function to have a known value. For example:

h1 = enthalpy(STEAM, T=T1, P=P1)

will return the value of h1 corresponding to known temperature and pressure, T1 and P1. If, however, the value of h1 is known, but T1 is unknown, the same equation will return the appropriate value of the temperature. Alternatively, the temperature could be found by:

T1 = temperature(STEAM, h = h1, P=P1)

The latter method is preferable in that the iterative calculations implemented for steam are less likely to have convergence difficulty.

The built-in thermophysical property functions are listed below in alphabetical order. The units, which depend on the choices made with the Unit System command in the Options menu, are shown in brackets. One or more examples, showing the allowable formats for the function, are also given.

Conductivity [W/m-K, Btu/hr-ft-R] returns the thermal conductivity of the specified substance. For substances modeled as an ideal gas, the conductivity function takes temperature as its only argument in addition to the substance name. Real fluids require temperature and pressure as arguments. Steam, Water and Steam_NBS will accept specific volume as an alternative to pressure. For AIRH2O (moist air), the temperature, pressure, and humidity ratio (or relative humidity) must be supplied.

Examples: k1 = conductivity (AIR, T=200)
k2 = conductivity (AMMONIA, T=100, P=200)
k3 = conductivity (STEAM_NBS, T=100, v=0.345)
k4 = conductivity (AIRH2O, T=80, P=14.7,R=0.5)

Density [kg/m^3, kgmole/m^3, lb/ft^3, lbmole/ft^3] returns the density of a specified substance. Two arguments are required for all pure substances; three are needed for moist air.

Example: d1 = **Density**(AIR, T=300, P=100)
d2 = **Density** (Steam, h=850, P=400)
d3 = **Density** (AirH2O, T=70, R=0.5, P=14.7)

DewPoint [°F, °C, R, K] returns the dewpoint temperature for air-water gas mixtures. This function can be used only with AIRH2O as the substance name. Three arguments follow the substance name in any order: temperature, total pressure, and relative humidity (or humidity ratio or wetbulb temperature).

Example: D1 = dewpoint(AIRH2O, T=70, P=14.7, w=0.010)
D2 = dewpoint(AIRH2O, T=70, P=14.7, R=0.5)
D3 = dewpoint(AIRH2O, T=70, P=14.7, B=50)

Enthalpy [kJ/kg, kJ/kgmole, Btu/lb Btu/lbmole] returns the specific enthalpy of a specified substance. The exact form of the enthalpy function depends on the substance and independent variable(s) selected. Substances which obey the ideal gas law, such as air, require a single argument, (temperature or internal energy), in addition to the substance name whereas real fluid substances, e.g., STEAM and CARBONDIOXIDE, will always require two independent variables. For AIRH2O, three arguments are required.

Example: h1 = enthalpy(AIR, T=300)
h2 = enthalpy(STEAM, T=900, P=300)
h3 = enthalpy(AIRH2O, T=70, P=14.7, R=0.50)

Entropy [kJ/kg-K, kJ/kgmole-K, Btu/lb-R, Btu/lbmole-R] returns the specific entropy of a specified substance. For all pure substances, the entropy function always requires two arguments, in addition to the substance name. For AIRH2O, three arguments are required.

Example: s1 = entropy(O2, T=400, P=100)
s2 = entropy(AIRH2O, T=70, P=14.7, R=0.50)

HumRat [dimensionless] returns the humidity ratio (defined as the mass of water vapor per mass of dry air) for air-water gas mixtures. This function is applicable only to the substance AIRH2O. The function requires three arguments which must include pressure and any two remaining independent variables such as temperature, relative humidity, enthalpy, or dew point.

Example: w1 = humRat(AIRH2O, T=70, P=14.7, R=0.50)
w2 = humRat(AIRH2O, T=70, P=14.7, h=25)

IntEnergy [kJ/kg, kJ/kgmole, Btu/lb, Btu/lbmole] returns the specific internal energy of a specified substance. The exact form of the **IntEnergy** function depends on the substance and independent variable(s) selected. Substances which obey the ideal gas law, such as air, require a single argument (temperature or enthalpy) whereas real fluid pure substances, like steam, will always require two arguments in addition to the substance name. AIRH2O requires three additional arguments.

Example: u1 = intEnergy(AIR, T=300)
u2 = intEnergy(STEAM, T=1320, P=300)
u3 = intEnergy(AIRH2O, T=70, P=14.7, R=0.50)

MolarMass returns the molar mass (often called molecular weight) of the fluid provided as the parameter.

Example: M_CO2 =MolarMass(CarbonDioxide)

Pressure [kPa, bar, psia, atm] returns the pressure of a specified substance. The argument list for the pressure function always requires the substance name followed by two arguments, each item separated by commas. The pressure function is not implemented for AIRH2O; however, an unknown pressure can still be determined using any of the functions which are applicable to moist air and which take pressure as an argument.

Example: P1 = pressure(STEAM, h=1450, T=900)

P_Crit [kPa, bar, psia, atm] returns the critical pressure of the specified fluid. The fluid may be a fluid name or a string variable. Critical property information is not available for ideal gas substances.

Example: Pc=P_Crit(R134a) "returns the critical pressure of R134a"

Quality [dimensionless] returns the quality (vapor mass fraction) for substances modeled as real fluids such as WATER and R12. Two independent arguments are required. Temperature and pressure are not independent for saturated states. If the state of the substance is found to be subcooled, the quality is returned as –100. If it is superheated, 100 is returned.

Example: x1 = quality(R12, h=50, T=80)

Relhum [dimensionless] returns the relative humidity as a fractional number for air-water gas mixtures. There are three arguments to this function, in addition to the substance name, AIRH2O. The three arguments are temperature, total pressure and any two remaining independent variables such as temperature, wetbulb, enthalpy, dew point, or humidity ratio.

Example: R1 = relhum(AIRH2O, T=70, P=14.7, w=0.01)
R2 = relhum(AIRH2O, T=70, P=14.7, h=25)
R3 = relhum(AIRH2O, T=70, P=14.7, B=55)

Specheat [kJ/kg-K, kJ/kgmole-K, Btu/lb-R, Btu/lbmole-R] returns the constant pressure specific heat of the specified substance. For pure substances which obey the ideal gas law, the specific heat function has temperature as its only other argument in addition to the substance name. The temperature and pressure must both be provided as arguments for substances modeled as real fluids. The specific heat of the liquid or vapor may be returned, depending on the temperature and pressure values provided.

Example: Cp1 = specheat(AIR, T=350)
Cp2 = specheat (AMMONIA, T=100, P=30)

Temperature [°C, K, °F, R] returns the temperature of the substance. The exact form of the function depends on the substance and argument(s) selected. Substances which are assumed to obey the ideal gas law, such as air, may require one or two arguments whereas pure real fluid substances, like STEAM, will always require two arguments.

Example: T1 = temperature(AIR, h=300)

T2 = temperature(AIR, s=1.75, P=100)

T_Crit [°C, K, °F, R] returns the critical temperature of the specified fluid. Critical property information is not available for ideal gas substances.

Example: Tc=T_Crit(R134a)

Volume [m^3/kg, m^3/kgmole, ft^3/lb, ft^3/lbmole] returns the specific volume of a specified substance. Two arguments are required for all pure substances; three are needed for moist air.

Example: v1 = **Volume**(AIR, T=300, P=100)
v2 = **Volume**(Steam, h=850, P=400)
v3 = **Volume**(AirH2O, T=70, R=0.5, P=14,7)

V_Crit [m^3/kg, m^3/kgmole, ft^3/lb, ft^3/lbmole] returns the critical specific volume of the specified fluid. Critical property information is not available for ideal gas substances.

Example: vc=v_Crit(R134a) "returns the critical volume of R134a"

Wetbulb [°C, K, °F, R] returns the wetbulb temperature for air-water gas mixtures. This function is applicable only to the substance AIRH2O. There are three arguments to this function, in addition to the substance name. The three arguments are temperature (or enthalpy), total pressure, and relative humidity (or humidity ratio or dewpoint).

Example: B1 = wetbulb(AIRH2O, T=70, P=14.7, w=0.01)
B2 = wetbulb(AIRH2O, h=25, P=14.7, w=0.01)
B3 = wetbulb (AIRH2O, h=25, P=14.7, D=30)

Viscosity [N-sec/m^2, lb_m/ft-hr] returns the dynamic viscosity of the specified substance. For substances modeled as an ideal gas, the viscosity function takes temperature as its only argument in addition to the substance name. Temperature and pressure arguments are required for real fluids. STEAM and STEAM_NBS will accept temperature and specific volume. Relative humidity or humidity ratio must be supplied for AIRH2O.

Example: v1 = viscosity(AIR, T=300)
v2 = viscosity(R12, T=40, P=30)
v3 = viscosity(STEAM_NBS, T=100, v=0.335)
v4 = viscosity(AIRH2O, T=80, P=14.7, R=0.5)

Using Lookup Files and the Lookup Table

A Lookup file is a two-dimensional set of data with a specified number of rows and columns. Lookup files provide a means to enter functional relationships with tabular data and to use these relationships in the solution of the equations. Lookup files can be stored in a disk file. Alternatively, a single Lookup file, called the Lookup Table, can exist in the Lookup Table Window. The six menu commands which pertain to the Lookup Table Window appear at the bottom of the Options menu and are summarized here.

New Lookup creates a new empty Lookup Table with a specified number of rows and columns in the Lookup Table Window. If a Lookup Table already exists in the Lookup Table window, it will be overwritten.

Open Lookup reads a Lookup file from the disk into the Lookup Table Window. If a Lookup Table already exists in the Lookup Table window, it will be overwritten. There are two lookup file formats identified with filename extensions of .LKT and .TXT. Either type can be opened with the Open Lookup command. The .LKT format is a binary format unique to EES that is created with the Save Lookup command. The .TXT format is an ASCII text file that can be created with the Save Lookup command in EES or with any other text processing application.

<u>Binary Lookup files (.LKT)</u>
Binary Lookup files store all of the information that appears in a Lookup Table window in a binary file on disk, including the data, and the column name, units, and display format for each file type. A binary (.LKT) Lookup file is created using the Save Lookup Table command. Once created, the Lookup file can be opened into the Lookup Table window using the Open Lookup Table command. Binary files require less disk storage space and the can be opened and saved more quickly by EES. However, they cannot be created, edited or viewed by any application other than EES.

<u>ASCII Lookup files (.TXT)</u>
There are several variations for the ASCII Lookup file format. In the basic form, the first line of the file contains the number of rows and columns in the table. Each following line provides the data for one row with the value for each column separated by one or more spaces or a tab character. The basic form does not provide a means of specifying the names, units, or display format for the data. EES assigns the names "COLUMN1', 'COLUMN2', etc. and these column names should be used when the file is used with the **Interpolate** or **Differentiate** commands. Automatic formatting is used to display the data if the file is read into the Lookup Table with the Open Lookup Table command. The following example shows the ASCII data needed for a Lookup file with five rows and three columns.

```
5   3
1       11      111
2       22      222
3       33      333
4       44      444
5       55      555
```

If a negative number is provided in the file for the number of rows, EES will determine the number of rows of data in the file. If the number of columns is a negative number, EES will expect to find the format specification (e.g., A3, F3 or E4) followed by one space and then the column heading and units for each column. The units are enclosed in square brackets. The following lines contain the data for each row, separated by one or more spaces or a tab. The example below would create a table with 2 rows and 3 columns. The columns would be formatted with E4, F0, and F3 format specifications and the column names will be ColA, ColB, and ColC.

```
2    -3
E4 ColA [Btu]
F0 ColB
F3 Col
1.23E-12      2       4.56
2.34E-11      4       7.89
```

In addition to being read into the Lookup Table, Lookup files in either the binary or ASCII formats can be accessed directly with **Interpolate**, **Differentiate**, **Lookup**, **LookupCol**, and **LookupRow** functions. These functions are documented below.

Save Lookup saves the Lookup Table in the Lookup Table Window as a Lookup file on the disk. Lookup files can be accessed with the Lookup functions described below. The Lookup file can be saved as a binary file with a .LKT filename extension or as an ASCII text file. The ASCII format allows the data to be exported to another application. Note that the content of the Lookup Table Window is also saved with other problem information when the Save command is issued. It is not necessary to save the Lookup Table separately unless it is to be used by more than one EES program or the program requires more than one Lookup table.

Insert/Delete Rows allows the number of rows in the Lookup Table to be changed.

Insert/Delete Cols allows the number columns in the Lookup Table to be changed.

Delete Lookup deletes the Lookup Table and recovers the memory it required.

Data in the Lookup Table can be accessed with the **Differentiate**, **Interpolate**, **Lookup**, **LookupRow**, and **LookupCol** functions. These functions may either operate on the data in the Lookup Table Window or in a Lookup file on disk. In the latter case, the filename must be provided as the first argument of the function. The file name can be supplied as a string constant (surrounded with single quotes) or as a string variable, identified by the $ character at the end of the variable name. Chapter 7 provides details on the use of string variables. The filename extension can either be .LKT (for binary lookup files) or .TXT (for ASCII lookup files). If a filename extension is not provided, EES will assume that the file format to be binary (i.e., EES will automatically append the .LKT extension).

Differentiate('*Filename*', 'ColName1', 'ColName2', ColName2=Value**)** returns the derivative determined from two columns of tabular data based on cubic interpolation. These data can be in the Lookup table, a Lookup file, or the Parametric table. 'FileName' is an *optional* parameter which, if supplied, must be the name of an existing Lookup file having either a .LKT or .TXT filename extension previously stored with the Save Lookup Table command. The filename can be entered either a string constant (within single quotes) or as a string variable that has been assigned to the name of the file. If the 'FileName' parameter is not supplied, the **Differentiate** function will be applied on the existing Lookup Table.

ColName1 and ColName2 are the column header names. The single quotes enclosing the column header names are optional. These columns names can also be supplied with string variables. The final parameter is of the form ColName2=Value where the text to the left of the equal sign can be either of the column header names (ColName1 or ColName2) specified with the two previous parameters. Value is a numerical value or expression. EES will return an estimate of the derivative d(ColName1)/d(ColName2) at a point fixed by the specified value of either ColName1 or ColName2. The **Differentiate** function can also be made to operate on data in the Parametric table if Filename is set to 'Parametric'. In this case, the values in the Parametric table must already exist, such as those entered from the keyboard. Values which are to be calculated when the Solve Table command is issued cannot be used with the **Differentiate** command.

Examples: dXdY=Differentiate('X', 'Y', Y=2.34) {returns the derivative dX/dY at a value of Y=2.34 using data in the Lookup Table. Quotes are optional}

Y=Differentiate('C:myFile',T,X,T=100) {returns the derivative dT/dX at a value of T=100 using data from Lookup file myFile.LKT on drive C}

Interpolate('*Filename*', 'ColName1', 'ColName2', ColName2=Value**)** returns an interpolated or extrapolated value from tabular data in the Lookup table, a Lookup file, or the Parametric table using cubic interpolation. 'FileName' is an optional parameter. It can be supplied as a string constant (within single quotes) or as a string variable. If this parameter is not supplied, the **interpolate** function will applied on the existing Lookup Table. If 'FileName' is supplied, it must be the name of an existing Lookup file having either a .LKT or .TXT filename extension. ColName1 and ColName2 are the column header names. These single quotes enclosing the column header names are optional. The column header names can alternatively be entered as string variables. The final parameter is of the form ColName2=Value where the text to the left of the equal sign can be either of the column header names (ColName1 or ColName2) specified with the two previous parameters. Value is a numerical value or expression. EES will return the interpolated value from the data in column ColName1 corresponding to the specified value of ColName2. If the value of ColName1 is supplied, EES will return the interpolated value of ColName2. If Filename is 'Parametric', the **interpolate** command will be applied to the existing Parametric table. In this case, the values in the Parametric table must already exist, such as those entered from the keyboard. Values which are to be calculated when the Solve Table command is issued cannot be used with the **interpolate** command.

Examples: Z=interpolate('Col1', 'Col2', Col1=2.3) {returns a value from column Col2 of the Lookup table corresponding to a value in Col1 equal to 2.3 using cubic interpolation. Note that the quotes are optional.}

X= Interpolate(C:\myData,X,Y,Y=4.5) {returns a value from column X in the lookup table called myData.LKT on drive C: corresponding to a value in column Y equal to 4.5 using cubic interpolation.}

interpolate1(*'Filename'*, 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses linear interpolation.

interpolate2(*'Filename'*, 'ColName1', 'ColName2', ColName2=Value**)** provides the same function as the **interpolate** command except that it uses quadratic interpolation.

Lookup(*'Filename'*, Row, Column**)** returns the value in the Lookup Table or Lookup file at the specified row and column. Note that the column can be specified either by providing a numerical value (or expression) for the column number or by providing the column name as a string constant (enclosed in single quotes) or as a string variable. An older format in which the column name is preceded with the # symbol is still supported. 'Filename' is optional. If a filename is provided, EES will first check to ensure that the specified Lookup file exists and then load the Lookup file data into memory. The row and column arguments need not be integers. The value returned will be interpolated between rows and columns as needed. For example, Lookup(2.5, 3) will return a value which is midway between the values on the second and third rows in the third column. If the specified row or column is less than 1, the value in the first row or column will be returned. Similarly, if the row or column is greater than the number of rows and columns in the lookup table, the value in the last row or column will be returned. The **Lookup** function can be used with the **LookupCol** and **LookupRow** functions to provide interpolated values of user-supplied tabular information. However, the **Interpolate** commands are usually more convenient for this purpose.

Examples:
X=Lookup(1,2) { Set X to the value in row 1, column 2 of the Lookup table}
X=Lookup(1,'X') { Set X to the value in row 1 of the column in the Lookup table which is named X. }
X=Lookup('C:\abc\ CopperK,R,'T') {Set X to the value in row R and the column which is named T in Lookup file C:\ abc\ CopperK.LKT}

LookupCol(*'Filename'*, Row, Value**)** uses the data in the specified row of the Lookup Table or Lookup file to determine the column which corresponds to the value supplied as the second argument. The column value returned may not be an integer. Interpolation between columns will be provided as needed. The purpose of the **LookupCol** function is to provide a means of relating tabular information in different rows of the Lookup Table or Lookup file.

Examples:
C=LookupCol(2, 100) {Set C to the column number in row 2 of the Lookup table which has a value of 100}
C=LookupCol('C:\abc\ CopperK', R, X) {Set C to the column number in row R of \Lookup file C:\abc\ CopperK.LKT having the value X}

LookupRow(*'Filename'***,** Column, Value**)** uses the data in the specified column of the Lookup Table or Lookup file to determine the row corresponding to the value supplied as the second argument. Note that the column can be indicated either by supplying its numerical value or by providing the column name as a string constant (enclosed in single quotes) or as a string variable. An older format in which the column name is preceded by the # symbol is also accepted. The row value returned may not be an integer. Interpolation between rows will be provided as needed. The purpose of the **LookupRow** function is to provide a means of relating tabular information in different columns of the Lookup Table.

Examples:
R=LookupRow(2, 100) {Set R to the row number in column 2 of the Lookup table which has a value of 100}
R=LookupRow('C:\abc\ CopperK', C, X) {Set R to the row number in column C Lookup file C:\abc\ CopperK.LKT which has the value X}

When a new Lookup Table is created, the columns are initially named **Column1**, **Column2**, etc. These default names and the table display format can be changed by clicking the left mouse button in the header cell as indicated in the Lookup Window section of Chapter 2.

Information can be copied to or from the Lookup Table via the Clipboard. In this way, data may be transferred between the Lookup Table and the Parametric Table or between other applications such as a spreadsheet program. To select a rectangular group of cells in the Table, click the left mouse in the upper left cell. Hold the Shift key down and then click in the lower right cell. Selected cells will be displayed in inverse video. Use the Select All command in the Edit menu to select all of the cells in the Lookup table. Next, use the Copy command in the Edit menu to copy a selected range of table cells *to* the Clipboard. Hold the Ctrl key depressed if you wish to copy the column header name and units, in addition to the selected data. Data may be copied *from* the Clipboard by clicking the upper-left cell into which the data are to be pasted, followed by the Paste command. The data in the Clipboard will be pasted into the Lookup Table, starting from the selected cell.

CHAPTER 5

EES Functions, Procedures and Modules

Most high-level programming languages allow the user to write subprograms. EES also offers this capability in a variety of ways. An EES subprogram is a function, procedure, or module written within EES. A function is a subprogram that accepts one or more inputs and returns a single result. A procedure can return one or more results. A module is similar to a procedure in that it can return one or more results. However, it differs from a procedure in that it employs equalities rather than assignment statements, as explained below. EES can access both internal subprograms that have been written within EES and external subprograms written in Pascal, C, C++, FORTRAN, or any compiled language. The development of external subprograms is described in Chapter 6. Both internal and external subprograms can be stored in the USERLIB\ subdirectory from which they are automatically loaded when EES is started.

EES subprograms offers a number of advantages. First, they make it easier to formulate the solution for a complicated system by breaking the problem up into a number of smaller parts. Programs which rely on subprograms are easier to understand. Second, subprograms can be saved in a library file and reused in other EES programs. Third, EES functions and procedures (but not modules) allow use of *if then else, repeat until* and *goto* statements. The statements appearing in functions and procedures differ from those in the main body of EES in that they are assignment statements, similar to those used in most high-level programming languages, rather than equality statements. They are executed in the order in which they appear. Modules employ equality statements just as those used in the main body of an EES program. EES reorders the equality statements as needed to efficiently solve the equations. The combination of both statement types offers a great deal of flexibility in the manner in which a problem can be formulated in EES.

There are several ways to access subprograms. The Merge command in the File menu can be used to import EES subprograms from one EES file into another. In addition, EES also allows subprograms to be saved in a library file. Library files are EES files containing one or more functions, procedures, and/or modules which have been saved with a .lib filename extension using the Save As command. Subprograms stored in library files that reside in the USERLIB\ subdirectory are automatically and transparently loaded when EES starts. Library files can also be loaded with the Load Library command in the File menu and with the $INCLUDE directive. Functions, procedures and modules in library files act just like EES internal functions. They can even provide help when requested. The steps necessary for creating library files is described at the end of this chapter.

EES Functions

EES provides the capability for the user to write functions directly in the Equations window using the EES equation processor. EES functions are similar to those in Pascal. The rules for these functions are as follows:

1. The user functions must appear at the top of the Equations window, before any modules or any equations in the main body of the EES program.
2. User functions begin with the keyword FUNCTION. The function name and arguments, enclosed in parentheses and separated by commas, follow on the same line.
3. The function is terminated by the keyword END.
4. The equations appearing in EES functions and procedures are fundamentally different from those appearing in the main body of EES. The equations in functions and procedures are more properly called assignment statements, similar to those used in FORTRAN and Pascal. An assignment statement sets the variable identified on the left of the statement to the numerical value on the right. X:=X+1 is a valid assignment statement but it obviously cannot be an equality, as assumed for all equations in the main body of EES. The := sign (rather than the = sign) is used to signify assignment. However, EES will accept an equal sign in assignment statements if the ☒ **Allow = in Functions/Procedures** control in the Display Options dialog window (Options menu) is selected.
5. EES normally processes the assignment statements in a function or procedure in the order they appear. However, *If Then Else*, *Repeat Until* and *goto* statements may be used in functions and procedures to alter the calculation order. The format of these logic control statements is described below.
6. Functions are called simply by using their name in an equation. The arguments must follow the name, enclosed in parentheses. The function must be called with the same number of arguments appearing in the FUNCTION statement.
7. Equations in user functions may call any of the built-in functions. In addition, they may call any previously-defined user function or procedure or any functions or procedures previously loaded as Library files. Recursive functions which call themselves are, however, not allowed. Functions may not call modules.
8. All variables used in the function body are local to the function except those variables defined in the scope of the $COMMON directive. The function returns the value to which its name is assigned.
9. Functions always operate in real mode regardless of the setting for complex algebra.

Functions can be used to implement an analytical relationship between two or more variables. For example, the specific availability of a flowing stream, often called ψ, is

$$\psi = (h - h_o) - T_o\,(s - s_o) + V^2/2 + g\,z$$

where

h and s are specific enthalpy and entropy, respectively
h_o and s_o are specific enthalpy and entropy at the ‘dead’ state condition, T_o and P_o
V is the velocity
g is gravitational acceleration
z is elevation, relative to a selected zero point

Once the temperature and pressure of the dead state are selected, h_o and s_o are constants. A user function for the availability of steam, with T_o=530 R and P_o=1 atm, could be implemented by placing the following statements at the top of the Equations window. A reference to psi(T1, P1, V1, Z1) from an equation would return the specific availability of steam in Btu/lb_m for the chosen ‘dead’ state.

```
FUNCTION psi(T, P, V, Z)
    To := 530      “R    dead state temperature”
    ho := 38.05    “Btu/lbm    specific enthalpy at dead state conditions”
    so := 0.0745  “Btu/lbm-R    specific entropy at dead state conditions”
    h := enthalpy(STEAM, T=T, P=P)
    s := entropy(STEAM, T=T, P=P)
    g = 32.17     “ft/s^2  gravitational acceleration
    psi := (h-ho)- To * (s – so) + (V^2 / 2 + g * Z) * Convert(ft^2/s^2, Btu/lbm)
END
```

Functions can also be used to change the name of any built-in function and/or to shorten the argument list. For example, the following function changes the name of **humrat**, the built-in function for humidity ratio, to **w**, eliminates the need to specify the substance AIRH2O as an argument, and sets the total pressure to 100 kPa in each case.

```
FUNCTION w(T,RH)
    w := humrat(AIRH2O, T=T, P=100, R=RH);
END
```

The two example functions both employ EES internal property functions and as a result, they depend on the EES unit settings to be properly set. Using the **UnitSystem** function (Chapter 4) and the IF THEN ELSE statements documented below, it is possible to write general functions that will operate correctly with any unit settings.

EES Procedures

EES procedures are very much like EES functions, except that they allow multiple outputs. The format of a Procedure is:

```
PROCEDURE test(A,B,C : X,Y)
...
...
X :=...
Y :=...
END
```

Procedures must be placed at the top of the Equations window, before any of the modules or equations in the main body of an EES program. The procedure name, TEST, in the example above, can be any valid EES variable name. The argument list consists of a list of inputs and a list of outputs separated by a colon. In the example above, A, B, and C are inputs and X and Y are outputs. Each procedure must have at least one input and one output. Each output variable must be defined by an equation with the output variables name on the left of the assignment sign. An END statement closes the procedure.

To use the procedure, place a CALL statement anywhere within your equations. The CALL statement appears as

```
...
CALL test(1,2,3 : X,Y)
...
```

The numbers of inputs and outputs in the CALL statement argument list must exactly match the PROCEDURE declaration statement. The arguments may be constants, string variables, numerical variables, or algebraic expressions. Additional arguments can be passed between the main body of an EES program and a procedure using the $COMMON directive. EES will evaluate the outputs using the input variables supplied in the argument list. Procedures may also call other functions and procedures, provided that they are defined previously. Procedures may not call modules.

The equations within a procedure differ from ordinary EES equations in modules or in the main body of an EES program. First, all variables except for the inputs and outputs are local to the procedure. Second, the equations are really assignment statements, rather than equalities, and to make this distinction clear, the assignment symbol (:=) is used in place of the equal sign. You may override this convention by enabling the ☒ **Allow = in Functions/Procedures** control in the Preferences dialog window (Options menu). Third, *if then else*, *repeat until* and *goto* statements may be used. The format of these flow control statements is described in the next section.

Implicit equations can not be directly solved in a procedure or function, as they are in modules and in the main equation body. Using the *If Then Else*, *Repeat Until* and *goto* statements, it is possible to program your own iterative loop. However, it is also possible to have EES solve implicit equations within a procedure. For example, consider the following two non-linear equations.

```
X^3 + Y^2 = 66
X/Y = 1.23456
```

To solve for X and Y in a procedure, subtract the right-hand side from the left hand side of each equation and set them to residuals, R1 and R2, respectively. Now use EES to solve for X and Y such that the residuals are 0. Here's a program which does this. However, it should be noted that implicit equations could be solved more directly and efficiently using a module, as described later in this chapter.

```
PROCEDURE Solve(X,Y:R1,R2)
    R1:=X^3+Y^2-66
    R2:=X/Y-1.23456
END

CALL Solve(X,Y:0,0)    {X = 3.834, Y = 3.106 when executed}
```

Procedures offer a number of advantages for the EES user. Commonly-used procedures may be saved separately and merged into the Equations window with the Merge command in the File menu. Alternatively, the procedure could be saved as a library file so that it is loaded automatically when EES is started. Procedures can be selectively loaded with the Load Library command in the Options menu or with the $INCLUDE directive. For example, the equations describing a turbine can be entered once and saved. Each time a turbine calculation is needed, the CALL Turbine statement can be used to determine the turbine work and outlet state variables.

EES supports both internal and externally-compiled procedures. Internal procedures are entered directly at the top of the Equations window, as described in this section. Compiled procedures are written in a high-level language such as C, Pascal, or FORTRAN and called from EES. The CALL statement for both types of procedures is identical. See Chapter 6 for a detailed description of writing and using compiled functions and procedures.

Single-Line If Then Else Statements

EES functions and procedures support several types of conditional statements. These conditional statements can *not* be used in modules or in the main body of an EES program. The most common conditional is the *If Then Else* statement. Both single-line and multiple-line formats are allowed for *If Then Else* statements. The single-line format has the following form.

If (Conditional Test) *Then* Statement 1 *Else* Statement 2

The conditional test yields a *true* or *false* result. The format is very similar to that used in Pascal. Recognized operators are =, <, >, <=, >=, and <> (for not equal). The parenthesis around the conditional test are optional. Note that string variables (see Chapter 7) can be used in the condition test. The *Then* keyword and Statement 1 are required. Statement 1 can be either an assignment or a *GoTo* statement. The *Else* keyword and Statement 2 are optional. In the single-line format, the entire *If Then Else* statement must be placed on one line with 255 or fewer characters. The following example function uses *If Then Else* statements to return the minimum of its three arguments.[9]

```
Function MIN3(x,y,z)    { returns smallest of the three values}
   If (x<y) Then m:=x Else m:=y
   If (m>z) Then m:=z
   MIN3:=m
End

Y = MIN3(5,4,6)      { Y will be set to 4 when this statement executes}
```

The AND and OR logical operators can also be used in the conditional test of an *If Then Else* statement. EES processes the logical operations from left to right unless parentheses are supplied to change the parsing order. Note that the parentheses around the (x>0) and (y<>3) are required in the following example to override the left to right logical processing and produce the desired logical effect.

```
If (x>y) or ((x<0) and (y<>3)) Then z:=x/y Else z:=x
```

[9] Note the the built-in MIN function accepts any number of arguments so this function would not be needed.

Multiple-Line If Then Else Statements

The multiple-line *If Then Else* statement allows a group of statements to be executed conditionally. This conditional statement can be used in functions and procedures, but not in modules or the main body of an EES program. The format is as follows:

```
If (Conditional Test) Then
    Statement
    Statement
    ...
Else
    Statement
    Statement
    ...
EndIf
```

The *If* keyword, the conditional test, and *Then* keyword must be on the same line. The parentheses around the conditional test are optional. The statements which are to be executed if the conditional test is true appear on following lines. These statements may include additional *If Then Else* statements so as to have nested conditionals. An *Else* (or *EndIf*) keyword terminates this first group of statements. The *Else* keyword should appear on a line by itself, followed by the statements which execute if the conditional test is false. The *EndIf* keyword, which terminates the multiple-line *If Then Else* statement, is required and it must appear on a line by itself. The format is illustrated in the following example. Indentation is used to make the logic flow more clear. However EES ignores the blank spaces. Also upper and lower case are treated equally.

```
Function IFTest(X, Y)
    If (X<Y) and (Y<>0) Then
     A:=X/Y
     B:=X*Y
      If (X<0) Then          { nested If statement}
       A:=-A;  B:=-B
     EndIf
    Else
     A:=X*Y
     B:=X/Y
    EndIf
    IFTest:=A+B
End

G=IFTest(-3,4) { G will be set to 12.75 when this statement executes}
```

GoTo Statements

EES will normally process the assignment statements in a function or procedure in the order they appear starting with the first statement. However, the flow control can be altered using *GoTo* statements. The format of a *GoTo* statement is simply

```
GoTo #
```

where # is a statement label number which must be an integer number between 1 and 30000. Statement labels precede an assignment statement separated with a colon (:). The *GoTo* statement must be used with *If Then Else* statements to be useful. The following function illustrates the use of *GoTo* and *If Then Else* statements in the calculation of the factorial of a value supplied as the argument.

```
Function FACTORIAL(N)
    F:=1
    i:=1
10: i:=i+1
    F:=F*i
    If (i<N) Then GoTo 10
    FACTORIAL:=F
End

Y= FACTORIAL(5)   { Y will be set to 120 when this statement executes}
```

Repeat Until Statements

Looping within functions and procedures can be implemented with *If Then Else* and *GoTo* statements described above, but it is generally more convenient and readable to use a Repeat Until construct. The *Repeat Until* statement has the following format. Note that *Repeat Until* statements can only be used in functions and procedures.

```
Repeat
    Statement
    Statement
    ...
Until (Conditional Test)
```

The conditional test yields a *true* or *false* result using one of the following operators: =, <, >, <=, >=, and <> (for not equal). The format is identical to that used in Pascal. Here is the same Factorial example presented in the previous section implemented with a Repeat Until construct.

Function **Factorial**(N)
F:=1
Repeat
F:=F*N
N:=N-1;
Until (N=1)
Factorial:=F
End

Y= **FACTORIAL**(5) { Y will be set to 120 when this statement executes}

Error Procedure

The Error procedure allows the user to halt calculations if a value supplied to a function or procedure is out of range. The format of the Error procedure is

Call Error('error message',X) or *Call* Error(X)

where 'error message' is an optional character string enclosed within single quotes and X is the value of the parameter which caused the error. If the error message string is not provided, EES will generate the following error message when it executes the ERROR procedure.

Calculations have been halted because a parameter is out of range. The value of the parameter is XXX.

The value of X supplied to the Error procedure replaces XXX. If an error string is provided, EES will display that string, inserting the value of X in place of the characters XXX. If a formatting option, such as F1 or E4 follows the XXX, as in the example below, the value of X will be accordingly formatted, otherwise a default format will be applied. The ERROR procedure will most likely be used with an IF - THEN - ELSE statement as in the following example.

Function abc(X,Y)
if (x<=0) then CALL ERROR('X must be greater than 0. A value of XXXE4 was supplied.', X)
abc:=Y/X
end

g:=abc(-3,4)

When this function is called, the following message will be displayed and calculations will stop: *X must be greater than 0. A value of -3.000E0 was supplied.*

Modules

Modules can be considered to be stand-alone EES subprograms that can be called from the main EES program. The format of a Module is similar to that for an internal Procedure. The Module is supplied with inputs and it calculates outputs. The formal format of the Module statement uses a colon in the argument list. The number of arguments to the provided to the left of the colon is the number of degrees of freedom in the module, or stated another way, the number of values that must be supplied to have the number of equations equal to the number of unknowns within the module. An example of a Module statement is

MODULE Testme(A, B : X, Y)

In this case, EES understands that there are two inputs (A and B) and two outputs (X and Y). However, EES Modules employ equalities rather than assignment statements as used in Procedures. In most cases, it does not matter what variables are specified as inputs as long as the appropriate number are specified. As a consequence, the colon which separates inputs from outputs is irrelevant and it can be replaced with a comma (or semi-colon for the European numerical format). The following form for the Module statement is equivalent to the format shown above.

MODULE Testme(A, B, X, Y)

A module is accessed with a CALL statement. For example, the following statement would access the Testme module.

CALL Testme(77,1.5, X,Y)

Note that if a colon is used to separate the inputs and output in the MODULE statement, it must also be used in the CALL statement. Similarly if a colon is not used in the MODULE statement, it should not be used in the CALL statement.

When EES encounters a CALL statement, it transparently grafts the equations in the module to the equations in the main program. The steps necessary for this process are as follows. First, every variable in the module, including the inputs and outputs in the MODULE statement, is renamed with a unique qualifier that EES can recognize. Then EES adds one equation for each input and output which sets the value of the parameter in the calling program to the value of the value in the module. Finally, all of the equations in the module, with their renamed variables, are merged into the EES program at the point at which it is called. If the module is called a second time, the process is repeated, but with a different qualifier for the variable names in the module. The net effect is that a copy of all of the equations in the module are merged into the main EES program each time a CALL statement is encountered. EES currently allows up to 6000 equations so rather large problems can be developed. EES then uses its efficient blocking techniques to reorganize the equations for optimal solution. As a result of this reorganization, the equations in the module may not necessarily be called in sequence. In fact, this is rarely the case. You can view the equations in the order that EES has rearranged them in the Residuals window. Equations from a module are identified with the module name followed by a backslash and then the call index number. For example, the following equation in the Residuals window

Turbine\2: h2=h1+Q/m

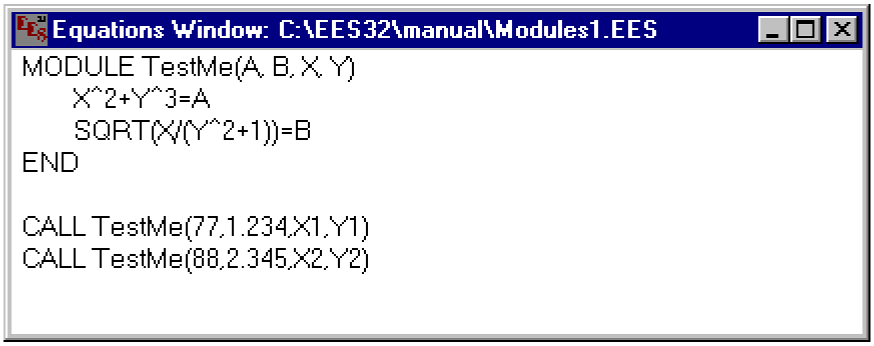
would indicate that the equation h2=h1+Q/m originated from the second call to the Turbine module.

The local values of variables in modules are normally not displayed in the Solution window. However, you can view these local solutions for modules which appear in your equations window by selecting the 'Show function/procedure/module variables' control in the Options tab of the Preferences dialog.

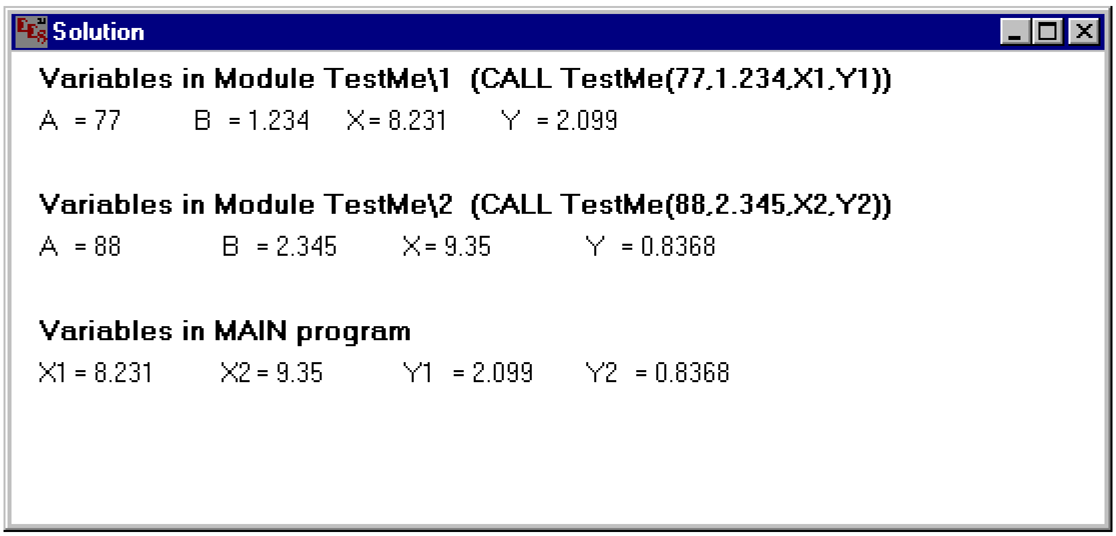
Variables values are normally passed to the module through the argument list. However, the $COMMON directive (Chapter 7) may be used to set the values of variables that are defined in the main program.

All variables defined within the module assume the same guess value, lower and upper bounds, and formatting information. The variable information is accessible with the Variable Info command. The local variables in a module are always real regardless of the Complex Numbers setting.

The Module is terminated with an END statement. A CALL statement is used to call the module, just as for the procedure. The important difference between a procedure and a module is that the module is composed of equality statements whereas the procedure is composed of assignment statements. Consequently, a module cannot support logic constructs such as IF THEN ELSE, but it can provide iterative solutions to implicit equations, when needed, and order independent equation input, just as in the main part of EES. Here is an example.



When **Solve** command is issued, EES will merge the equations in the Testme module twice into the main set of equations then solve the entire set. The Solution window will then appear as



Note that the local variables for each call to the Testme module will be shown only if the 'Show function/procedure/module values' control in the Options tab of the Preferences dialog is selected.

Modules may be stored as library files, just like internal functions and procedures. (Library files are described in the following section.) The library files can be automatically loaded if they are placed in the USERLIB subdirectory. Alternatively, a $INCLUDE directive can be used to transparently load a library file. Help can be included within the module using the same syntax as used in procedures or it can be supplied with a separate ASCII or Windows help file having the same filename as the library file with a .HLP filename extension.

Modules can significantly increase the capabilities of your EES programming.

Library Files

EES allows files containing one or more functions, procedures or modules (subprograms) to be saved as Library files. A Library file has a .LIB filename extension. When EES starts, it will automatically load all of the functions, procedures, and modules in the library files which reside in the USERLIB\ subdirectory. Library files can also be loaded manually with the Load Library command in the File menu and with the $INCLUDE directive. Library subprograms will not displayed in the Equations window. They are used just like EES built-in functions. To create a Library file, enter one or more functions, procedures, and/or modules into the Equations window. Compile the equations using Check, Solve or Solve Table. Then save the file with a .LIB filename extension using the Save As command.

Subprograms in library files can provide help information in the Function Info dialog window, just like the built-in functions. The help text is placed in the file as a comment in braces. The first character after the opening comment brace is a $, followed by the name of the function, procedure or module followed by a carriage return. The lines following, up to the closing comment brace, are the help text that will be displayed when the user selects the Info button in the Function Info dialog window as shown in the example below.

The Library file concept is among the most powerful features of EES because it allows the user to easily write customized subprograms for personal use or for use by others. The following example uses a library file to provide a fourth-order Runge-Kutta numerical integration function in EES. The Runge-Kutta algorithm is used to numerically solve a differential equation of the form:

$$\frac{dY}{dX} = f(X,Y)$$

where *f(X,Y)* is any function involving the dependent variable *Y* and independent variable *X*. *Y* must have a known initial value, Y0, corresponding to the initial value of *X*.

The Runge-Kutta algorithm has been implemented as a general purpose library function called RK4. RK4 requires 4 parameters: the initial value of X (LowX), the final value of X (HighX), the step size (StepX), and the value of Y at X=LowX (Y0). The function returns the value of Y at X=HighX. The RK4 function calls another function, fRK4(X,Y), to provide the value of dY/dX for given X and Y values. A dummy fRK4 function is provided in the RK4.LIB file as a placeholder. In an actual application, the user overrides the dummy fRK4 function by entering another fRK4 function in the EES Equations window. The RK4 and fRK4 functions have been saved in a library file called RK4.LIB in the USERLIB\ sub-directory. EES will load these functions when it starts. If you were to open the RK4.LIB file in EES, you would see the following statements. Note how the functions provide help text as a comment with the $filename key. Help can also be provided in an ASCII file having a .HLP filename extension.

```
FUNCTION fRK4(X,Y)
{$fRK4
fRK4 is a user-supplied function to evaluate dY/dX.  This
function is used with the RK4 function to solve differential
equations with the Runge-Kutta method.  Enter a fRK4(X,Y)
function in the Equations window to evaluate dY/dY for your
problem.  See the RK4 function for additional information.}
    fRK4:=(Y+X)^2
END

FUNCTION RK4(LowX,HighX,StepX,Y0)
{ $RK4
RK4 is a general purpose function which solves a first-order
differential equation of the form dY/dX=fRK4(X,Y) using the
Runge-Kutta 4th order algorithm.  The RK4 function calls function
fRK4(X,Y) supplied by the user to evaluate dY/dX at specified values
of X and Y.  The user must supply the fRK4 function.

RK4 requires four input parameters.  LowX is the initial
value of independent variable X.  HighX is the final value
of independent variable X and StepX is the step size.  Y0 is
the value of Y when X is equal to LowX.}

    X := LowX
    Y := Y0;
    Tol := 0.1*StepX
10:
    IF (X>HighX-Tol) THEN GOTO 20
    k1 := fRK4(X,Y)*StepX
    k2 := StepX*fRK4(X+0.5*StepX,Y+0.5*k1)
    k3 := StepX*fRK4(X+0.5*StepX,Y+0.5*k2)
    k4 := StepX*fRK4(X+StepX,Y+k3)
    Y := Y+k1/6+(k2+k3)/3+k4/6
    X := X+StepX
    GOTO 10;
20:
    RK4:=Y
END
```

Suppose you wish to numerically solve the equation $\int_0^2 X^2 dx$ using the RK4 function.

You provide a function fRK4 to evaluate the integrand, which is X^2 in this case. Your function overrides the fRK4 function in the RK4 library file. Assuming RK4 was in the USERLIB\ sub-directory when EES was started, all that would be needed is:

```
FUNCTION fRK4(X,Y)
     fRK4:=X^2
END
V=RK4(0,2,0.1,0)
```

When you solve this problem, EES will display V=2.667 in the Solution window.

$COMMON Directive

The $COMMON directive provides a means for passing information from the main program to internal functions, procedures, and modules. Use of $COMMON provides an alternative to passing values as arguments. This directive is similar in concept to the COMMON statement in FORTRAN. It differs in that information flow is one-way. Variable values can be passed from the main program to the function or procedure. However, the function or procedure may not assign or alter these values.

The $COMMON directive must directly follow the FUNCTION, PROCEDURE, or MODULE declaration on a line by itself. Variables appearing in the $COMMON statement are separated with commas, as in the following example.

```
FUNCTION  TESTCOMMON(X)
$COMMON B,C,D {variables B,C, and D are from the main program}
     TESTCOMMON:=X+B+C+D
END
B=4;  C=5;  D=6
G=TESTCOMMON(3)
```

$COMMON should only be used with functions, procedures, and modules appearing in the Equations window. It should not be used with library functions.

$INCLUDE Directive

The $INCLUDE directive provides an automatic method for loading a library file or ASCII text file containing EES equations. The format is:

$INCLUDE FILENAME

FILENAME is the filename including the filename extension which can be one of .TXT, .LIB, .FDL, .DLF, or .DLP. The filename should also include the complete path name, e.g., C:\EESW\myDefn.TXT. However, if a path name is not provided, EES will look in the current directory. If EES is unable to find the file, it will provide the opportunity to browse so that you can locate it. The $INCLUDE statement must be on a line by itself, starting in column 1. It is best to place the $INCLUDE directives at the top of the Equations Window to ensure that the directive is processed before compilation of the equations is initiated.

<u>.TXT Files</u>
If the filename extension is .TXT, EES expects FILENAME.TXT to be an ASCII text file containing EES equations. Syntax errors can not be identified in the library file so care should be taken to ensure the equations are correct before saving the library file. EES will include these equations with others in the Equations window during compilation. However, the equations and the variables associated with these equations will be hidden. Nested use of the $INCLUDE directive is not supported so that the text file must not include any $INCLUDE statements.

Equations can also be entered from a file with the Merge command in the File menu. The difference between the Merge command and the $INCLUDE directive is that equations entered with the Merge command are placed directly into the Equations window as if they were typed and they remain visible. The text entered with the $INCLUDE directive will be hidden. Note that the speed of editing in the Equations window is reduced as the size of the text in the Equations window increases. Use of the $INCLUDE directive for very large problems can eliminate this problem on slower machines.

<u>Library Files</u>
If the filename extension is .LIB, .FDL, .DLF, or .DLP, EES will expect the file to be a library file of a type corresponding to the filename extension. EES internal functions, procedures. and modules are recognized with the .LIB extension, whereas external functions use a .DLF extension and external procedures use either the .FDL or .DLP extension. EES will automatically load the referenced library file if it is not already loaded. Note that library files can also be loaded automatically when EES is started by placing them in the USERLIB subdirectory or by applying the Load Library command.

C H A P T E R 6

Compiled Functions and Procedures

EES has an extensive library of built-in functions, but it is not possible to anticipate the needs of all users. A remarkable feature of EES is that the user can add (and later remove) functions and procedures written in any compiled language, such as Pascal, C, C++ or FORTRAN. These compiled routines may have any number of arguments. Functions return a single value whereas procedures may return multiple values. The compiled routines are used in exactly the same manner as internal EES subprograms. This capability gives EES unlimited flexibility and it is among its most powerful features.

Compiled functions and procedures are written as dynamic link library (DLL) routines under the Windows operating system[10]. Compiled functions are identified with a .DLF. There are two formats for compiled procedures identified by .DLP and .FDL filename extensions. When EES is started, it examines the files in the EES USERLIB\ subdirectory. Any files having a .DLF, .DLP, or .FDL filename extension are assumed to be compiled functions or procedures and they are automatically loaded. External routines can also be loaded using the Load Library command in the File menu or with the $INCLUDE directive. The function name to be referenced in EES equations is the filename (without the extension).

Compiled functions and procedures can (optionally) be setup to work with the Function Info command (Options menu) so that they provide an example and detailed help when it is requested. The following sections of this chapter provide detailed information and examples of compiled functions and procedures.

EES Compiled Functions (.DLF files)

Compiled functions can be written in C, C++, Pascal, or any language which can produce a dynamic link library (DLL). The function statement header, however, must have a specific format. To avoid having to set a fixed upper limit on the number of inputs, the input information to a compiled function is implemented as a linked list. The linked list record or structure consists of an extended precision value and a pointer to the next input. The last input

[10] Note that there are both 16 and 32-bit DLLs in the Windows Operating System and they are not interchangeable. The 16-bit version of EES can only use 16-bit DLLs and the 32-bit version can only use 32-bit DLLs. The 32-bit version of EES can be used with Windows 95 or NT, but not with Window 3.1. The 16-bit version will run under any Windows operating system, but it executes more slowly than the 32-bit version. Instructions for preparing both DLL types are provided in this chapter. If you are not sure as to whether you are using the 16-bit or 32-bit version of EES, start the program and examine the startup screen. The 32-bit version of EES will display (32-bit) on the same line as the version number and in the main window title.

points to nil. Some compiled languages, such as FORTRAN 77 do not support pointers so .DLF compiled functions cannot be written in these languages The .FDL format described in this chapter should be used in this case.

The compiled function should check that the number of inputs supplied in the linked list is equal to the number the function expects. (The PWF function example in the next section shows how this checking can be done.) Although the values of the inputs may be changed in the function, these changes are local and will be disregarded by EES. Only the function result will be used by EES. A skeleton listing of a compiled function written in Borland's Delphi 1.0 (16-bit) or Delphi 3.0 (32-bit) is as follows:

```
library XTRNFUNC;
{$N+}

type
      ParamRecPtr = ^ParamRec;
      ParamRec = record  { defines structure of the linked list of inputs }
              Value: extended;
               next: ParamRecPtr;
      end;

function FuncName (var S:string; Mode:integer; Inputs:ParamRecPtr): extended; export; stdCall; [11]
     begin
              ...
              FuncName:=Value; { Funcname must be extended precision }
     end;

exports FuncName;

begin
end.
```

The major concern is the function header. To be recognized by EES, the function name, called FuncName in the above example, must be the same as the filename. The function statement has three arguments.

S is a standard 255-character Pascal string. The first character contains the actual length of the string. S can be used for both input and output. If the first parameter provided in the EES function is a string (within single quotes), EES will pass this string to the external routine. If an error is encountered, S should be set in the external routine to an appropriate error message. If the length of S is not zero, EES will terminate calculations and display S as an error message.

[11] The **stdCall** keyword is required for Delphi 3.0 (32-bit). It is not needed for Delphi 1.0 (16-bit)

Mode is an integer set by EES. If Mode=–1, then EES is requesting that the function return in S an example of the function call. If Mode>=0, then the function should simply return the function value. Currently, EES does not use the return value of Mode.

Inputs is a pointer to the head of a linked list of input values supplied by EES. Each input consists of a value (extended precision) and a pointer to the next input, as indicated by the ParamRec structure. The function may have one or more inputs. The next field of the last input will be a null pointer (nil). The function should count the inputs to be sure that the number supplied is as expected and issue an error message in S if this is not the case.

A skeleton listing of a compiled function written in Borland's C++ follows:

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#define EXAMPLE=-1;
// Use extern "C" to prevent C++ name mangling
extern "C"
{
 long double far pascal _stdcall _export FUNCNAME
      (char* S, int Mode, struct ParamRec *FirstInput);
}
int far pascal LibMain  //DLL entry point for initiation
  (HINSTANCE hInstance, WORD wDataSeg, WORD cbHeapSize, LPSTR lpstrCmdLine)
 {
      if (cbHeapSize) UnlockData(0);
      return TRUE;
 }
int far pascal WEP(int nParam)  //Windows exit procedure - not needed in Borland C++
 {
       return TRUE;
 }
struct ParamRec
{
      long double                 value;
      struct ParamRec             *next;
};
long double far _export _stdcall pascal FUNCNAME(char* S, int Mode, struct ParamRec *FirstInput)
  {
      ….
      ….
      return (v);
}
```

Note that the pascal keyword must be provided to ensure that the calling parameters are in an order that will be understood by EES.

The PWF Compiled Function

EES does not have any internal economic functions. An economic function called the present worth factor (PWF)[12] has been added as a compiled function. PWF is the present worth of a series of N future payments which inflate at rate *i* per period accounting for the time value of money with a market discount rate per period of *d*. The equation for PWF is

$$PWF(N,i,d) = \sum_{j=1}^{N} \frac{\left(1+i\right)^{j-1}}{\left(1+d\right)^{j}} = \begin{cases} \dfrac{1}{d-i}\ 1-\left(\dfrac{1+i}{1+d}\right)^{N} & if\ i \neq d \\ \dfrac{N}{1+i} & if\ i = d \end{cases}$$

where
- N is the number of periods (e.g., years)
- i is the interest rate per period, expressed as a fraction
- d is the market discount rate per period, expressed as a fraction

A compiled function, called PWF, has been written to do this economic calculation. This function is stored in the PWF.DLF file on the EES disk. EES treats this compiled function just like any of its internal functions. Shown on the following pages is the complete listing for the PWF compiled function written in Borland's Delphi 3.0 (32-bit version).

Four other compiled functions are included with EES. These functions implement a generalized equation of state using Redlich-Kwong-Soave equation[13].

Compressibility(Tr, Pr, w**)** returns the compressibility of a gas, i.e., the ratio of the specific volume of the gas to the specific volume of an ideal gas at the same conditions. Tr is the reduced temperature, Pr is the reduced pressure, and w is the acentric factor. The third parameter is optional.

EnthalpyDep(Tr, Pr, w**)** and **EntropyDep(**Tr, Pr, w**)** return the dimensionless enthalpy and entropy departures, respectively. The dimensionless enthalpy departure is defined as (h[ideal]-h)/(R Tc). h[ideal]-h is the difference in enthalpy between an ideal gas and a real gas at the same temperature and pressure, R is the gas constant and Tc is the critical temperature. The dimensionless enthalpy departure is similarly defined as (s[ideal]-s)/(R).

FugCoef(Tr, Pr, w**)** returns the dimensionless fugacity coefficient (fugacity / pressure).

Do you need a function which EES doesn't include? Write your own. It's EESY!

---

12 Duffie, J.A. and Beckman, W.A., *Solar Engineering of Thermal Processes*, 2nd edition, J. Wiley and Sons, 1992, Chapter 11.

13 G. Soave, Chem. Eng. Science, Vol. 27, pp. 1197-1203, 1972

Listing of the PWF Compiled Function in Borland's Delphi 3.0

```
library PWFP;
uses
  SysUtils, Classes;
{$N+}

const doExample = -1;

type
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:extended;
    next:ParamRecPtr;
  end;

 function CountValues (P: ParamRecPtr): integer;
  var  N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
       N := N + 1;
       P := P^.next
    end;
    CountValues := N;
  end; {CountValues}

function PWF(var S:Shortstring; Mode:integer; Inputs:ParamRecPtr):extended; export; stdcall;
  var  P: ParamRecPtr; V: extended;

 function CountValues (P: ParamRecPtr): integer;
  var
   N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
       N := N + 1;
       P := P^.next
     end;
     CountValues := N;
  end; {CountValues}

 function PWFCalc: extended;
  var
    Periods, NArgs: integer;
    interest, discount: extended;
  begin
    PWFCalc:=0; {in case of error exit}
    S := '';
    P := Inputs;
    Periods := round(P^.value);
    if (Periods < 1) then begin
       S := 'The number of periods for the PWF function must be >0.';
       exit;
    end;
```

```
    P := P^.next;
    interest := P^.value;
    if (interest >= 1) or (interest < 0) then begin
      S := 'The interest rate is a fraction and must be between 0 and 1.';
      exit;
    end;
    P := P^.next;
    discount := P^.value;
    if (discount >= 1) or (discount < 0) then begin
      S := 'The discount rate is a fraction and must be between 0 and 1.';
      exit;
    end;
    if (interest <> discount) then
      PWFCalc := 1 / (discount - interest) * (1 - exp(Periods * ln((1 + interest) / (1 + discount))))
    else
      PWFCalc := Periods / (1 + interest);
  end; {PWF}

 begin
   PWF:=1;
   if (Mode = doExample) then begin
     S := 'PWF(Periods,Interest,Discount)';
     exit;
   end;
   if (CountValues(Inputs)<>3) then
     S := 'Wrong number of arguments for PWF function.'
   else begin
     PWF:=PWFCalc;
   end;
 end; {PWF}

 exports  PWF;

begin
  {no initiation code needed}
end.
```

When this Pascal code is compiled with the Borland Delphi 3.0, a dynamically-linked library routine is created. The compiler automatically generates a .DLL filename extension for the compiled code. EES must distinguish compiled functions from compiled procedures. It does this by the filename extension. Compiled functions must have a .DLF filename extension. Rename the compiled file so that it has a .DLF extension.

Access the external PWF function by a statement of the following form in your EES program.

```
P = PWF(Periods,Interest,Discount)
```

EES Compiled Procedures (.FDL and .DLP files)

EES compiled procedures are very similar to EES compiled functions. In either case, the user supplies the function or procedure in compiled form as a Windows dynamically linked library routine. The major difference between functions and procedures is that procedures can return one or more values, whereas a function returns a single value. Procedures are useful, for example, for thermodynamic property evaluations where multiple properties (e.g., volume, enthalpy, entropy, etc.) are to be determined given one set of independent variables (e.g., temperature and pressure).

External procedures are written as dynamic link libraries (DLL's). Be careful to provide 16-bit DLLs for the 16-bit version of EES and 32-bit versions for the 32-bit version. There are two formats for external procedures and they are identified to EES by their filename extension. The two formats differ in the manner in which EES exchanges information with the external routine. The .FDL format passes inputs and outputs in double precision floating point arrays which may contain up to 50 elements. The .DLP format passes inputs and outputs as linked lists (as in the .DLF functions) so there is no limit to the number of inputs and outputs. EES identifies the format by the filename extension which must be .FDL or .DLP. External procedures using arrays to hold the inputs and outputs must have a .FDL filename extension. This is the usual case when providing DLLs written in FORTRAN. C, C++, and Pascal procedures can use either format.

Compiled procedures are accessed from EES with the CALL statement which has the following format,

CALL procname('text', A, B : X, Y, Z)

where

procname is the name of the procedure

'text' is an (optional) text string that will be passed to the procedure. This text can either be a string constant enclosed in single quote marks or a string variable. (See Chapter 7).

A and B are inputs. There may be one or more inputs, separated by commas, appearing to the left of the colon. Inputs may be numerical constants, EES variable names, or algebraic expressions. String variables cannot be supplied as inputs.

X, Y, and Z are outputs determined by the procedure. There should be one or more outputs to the right of the colon, separated by commas. Outputs must be EES numerical variable names. String variables cannot be provided for outputs.

Note that the CALL statement used to access compiled functions is identical in format to the CALL statement used for internal EES Procedures.

The following two sections describe the .FDL and .DLP external procedure formats and provide a simple example which should serve as a model.

Compiled Procedures with the .FDL Format - a FORTRAN Example

The .FDL format is illustrated by the following FORTRAN subroutine fragments. The code differs slightly depending on whether 16-bit or 32-bit libraries are being compiled and which compiler is being used.

<u>16-bit .FDL library using the Microsoft FORTRAN 5.1 compiler</u>

```
SUBROUTINE MYPROC(S,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
INTEGER*2 MODE, NINPUTS, NOUTPUTS
REAL*8 INPUTS(50), OUTPUTS(50)
CHARACTER*255 S
…
OUTPUTS(1)=…
…
RETURN
END
```

<u>32-bit .FDL library using the Digital Visual FORTRAN 5.0 compiler</u>

```
SUBROUTINE MYPROC(S,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
!DEC$ATTRIBUTES ALIAS:'MYPROC' :: MYPROC
!DEC$ATTRIBUTES DLLEXPORT :: MYPROC
INTEGER(4) MODE, NINPUTS, NOUTPUTS
REAL(8) INPUTS(50), OUTPUTS(50)
CHARACTER(255) S
…
OUTPUTS(1)=…
…
RETURN
END
```

S is a null-terminated C-style character string containing 255 characters. If the first parameter in the EES Call statement is a text string (within single quotes), EES will pass this string to the external program in S. When EES calls the subroutine with `MODE` =-1, it is asking for an example of the calling sequence of this procedure from EES to be placed in S so that it can be displayed in the Function Info Dialog window. S is also used to return user-supplied error messages if necessary. If an error is detected in the subroutine, `MODE` should be set to a value greater than 0 to signal EES to terminate calculations. If S is defined, it will be displayed in the EES error message. In normal operation, `MODE` =0 and S need not be defined.

`NINPUTS` and `NOUTPUTS` are the number of inputs and outputs provided by EES. The routine should check to see if these agree with the expected number of inputs and outputs and return an error condition (`MODE`>0) is this is not the case. `INPUTS` and `OUTPUTS` are arrays of 50 double precision (REAL*8) values. EES will supply the values in the `INPUTS` array. Results calculated by the subroutine are placed in the `OUTPUTS`.

The external program must be compiled and linked as a dynamic link library (DLL) routine. The compiling procedure differs among different languages and compilers. To compile and link a FORTRAN external procedure called MYPROC as a DLL in the Microsoft FORTRAN 5.1 environment to produce the MYPROC.FDL 16-bit external EES procedure, you would enter:

```
fl /c /Aw /Gw MYPROC.FOR
link MYPROC, MYPROC.FDL,NUL,/NOD LDLLFEW, MYPROC.DEF
```

The compiler and linker options are defined in the Microsoft FORTRAN 5.1 manuals. A definition file, MYPROC.DEF, is required by the linker. This file has the following format.

```
LIBRARY MYPROC
DESCRIPTION MYPROC TEST FDL'
APPLOADER '__MSLANGLOAD'
EXETYPE WINDOWS 3.0
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE SINGLE
HEAPSIZE 1024
EXPORTS MYPROC @1
         WEP   @2 RESIDENTNAME
```

Creating a 32-bit DLL with Digital Visual Fortran 5.0 is most easily done within the Microsoft Developer Studio environment. A new project workspace is selected as a Dynamic Link Library. The FORTRAN sources file(s) are inserted into the workspace and compiled with the standard options. Note that the two !DEC$ATTRIBUTES directives should be included in the main program, as noted above. The output filename in the Link settings should be set to MYPROC.FDL where MYPROC is the name that will be used in the EES Call statement. Alternatively, the default filename MYPROC.DLL should be changed to MYPROC.FDL after building the dynamic link library project.

The simple FORTRAN program listed below provides the product, dividend, sum, and difference of two input values. This program should provide a model for writing external EES procedures in FORTRAN.

Listing of the FORTRAN MDASF Program

```
      SUBROUTINE MDASF(S,MODE,NINPUTS,INPUTS,NOUTPUTS,OUTPUTS)
C.  The following two lines are specific to Microsoft Power Station 4.0
        !MS$ATTRIBUTES ALIAS:'MDASF' :: MDASF
        !MS$ATTRIBUTES DLLEXPORT :: MDASF
C.  Replace INTEGER(4) with INTEGER*2 for a 16 bit DLL in the following line
      INTEGER(4) MODE, NINPUTS, NOUTPUTS
      REAL(8) INPUTS(25), OUTPUTS(25)
      CHARACTER(255) S
C.
      IF (MODE.EQ.-1) GOTO 900
      IF (NINPUTS.NE.2) GOTO 100
      IF (NOUTPUTS.NE.4) GOTO 200
C.  DO CALCULATIONS
      X=INPUTS(1)
      Y=INPUTS(2)
      IF (ABS(Y).LE.1E-9) GOTO 300
      OUTPUTS(1)=X*Y
      OUTPUTS(2)=X/Y
      OUTPUTS(3)=X+Y
      OUTPUTS(4)=X-Y
      MODE=0
      S=''C
      RETURN
100   CONTINUE
C.  ERROR:  THE NUMBER OF INPUTS ISN'T WHAT THIS SUBROUTINE EXPECTS
C.  NOTE: SET MODE>0 IF AN ERROR IS DETECTED.  IF S IS EQUAL TO A
C.  NULL STRING, THEN EES WILL DISPLAY THE MODE NUMBER IN AN ERROR
C.  MESSAGE.  IF S IS DEFINED, EES WILL DISPLAY THE STRING IN THE
C.  ERROR MESSAGE.  THE C AT THE END OF THE STRING INDICATES C-STYLE
C.  S='MDASF REQUIRES 2 INPUTS'C
      MODE=1
      RETURN
200   CONTINUE
      S='MDASF EXPECTS TO PROVIDE 4 OUTPUTS'C
      MODE=2
      RETURN
300   CONTINUE
      S='DIVISION BY ZERO IN MDASF'C
      MODE=3
      RETURN
900   CONTINUE
C.  PROVIDE AN EXAMPLE OF THE CALLING FORMAT WHEN MODE=-1
      S='CALL MDASF(X,Y:A,B,C,D)'C
      RETURN
      END
```

Compiled Procedures with the .DLP Format - a Pascal Example

The .FDL format described in the previous section was illustrated with FORTRAN, but it can be implemented in any compiled language. The .DLP calling format described in this section uses linked lists for inputs and outputs, and thereby is not suitable for use with FORTRAN. There is essentially no difference in efficiency between the two formats. Both are provided for backward compatibility and complete flexibility.

Compiled procedures using the .DLP format are very similar to compiled functions (.DLF files) described previously. The only difference is that a procedure must have, in addition to a linked list of input values, a linked list of output values. The calling sequence for a compiled Pascal procedure with the .DLP format has the following form

```
procedure procname (var S: string; Mode: integer; Inputs, Outputs: ParamRecPtr);
```

S, Mode, and Inputs are identical to their counterparts for the EES compiled functions. Outputs is a linked list of extended values which provides the results of the calculations to EES in the order in which they appear in the CALL statement.

Shown on the following page is a complete listing of an EES compiled procedure, called MDAS (an acronym for MyDearAuntSally) which provides the product, dividend, sum, and difference of two input values. (This is the same program used in the .FDL example.) The code checks to make sure that the number of inputs and outputs provided in the CALL statement are what the routine expects before it does the calculations and sets S to an error message if this is not the case.

Example Compiled Procedure (.DLP) in Borland's Delphi 2.0

```
library MDASP;

uses
  SysUtils, Classes;
{$N+}
const Example = -1;
type
  ParamRecPtr=^ParamRec;
  ParamRec=record
    Value:extended;
    next:ParamRecPtr;
  end;

 function CountValues (P: ParamRecPtr): integer;
  var  N: integer;
  begin
    N := 0;
    while (P <> nil) do begin
      N := N + 1;
```

```
      P := P^.next
    end;
    CountValues := N;
  end; {CountValues}

procedure MDAS(var S:Shortstring; Mode:integer;
      Inputs,Outputs:ParamRecPtr); export; stdcall;

 procedure MyDearAuntSally;
  var
    P1, P2: extended;
    P: ParamRecPtr;
  begin
    P := Inputs;
    P1 := P^.Value;
     P := P^.next;
   P2 := P^.value;
    P := Outputs;
    P^.Value := P1 * P2;
    P := P^.next;
    P^.Value := P1 / P2;
    P := P^.next;
    P^.Value := P1 + P2;
    P := P^.next;
    P^.Value := P1 - P2;
  end; {doCall}

 begin {MDAS}
   if (Mode = -1) then
     S := 'CALL MDAS(In1,In2:Out1,Out2,Out3,Out4)'
   else begin
    if (CountValues(Inputs) <> 2) then begin
       S := 'Wrong number of inputs for MDAS.';
       exit;
    end;
    if (CountValues(Outputs) <> 4) then begin
       S := 'Wrong number of outputs for MDAS.';
       exit;
    end;
     MyDearAuntSally;
     S:='';
    end;
 end; {MDAS}

 exports
   MDAS;

begin
  {no initiation code needed}
end.
```

Help for Compiled Functions and Procedures

The Function Info dialog (Options menu) has a INFO button which, when used, provides help explaining the use of the selected function. When the user clicks the INFO button, EES will look for a file with the name of the compiled routine and a .HLP extension. This file can either be an ASCII text file or a Windows .HLP file. This help file will be displayed if the file is found in the directory in which the external library file resides; otherwise a message will appear which states the help is not available for this item.

If an ASCII file is provided, it should be formatted so that each paragraph ends with a carriage return. Long lines which do not fit within the Help window will be broken and word-wrapped as needed. Blank lines and spaces can be used to make the text more clear.

Note that the Windows .HLP allows figures and formatting options to be used in the .HLP file and so it is a better way of providing help. The Window .HLP file can be composed within any word processor which produces a RTF file or more conveniently, using any of the commercial Help generating programs.

.

CHAPTER 7

Advanced Features

The advanced features in EES allow the program to work with string, complex, and array variables and solve simultaneous algebraic and differential equations. The commands and functions which implement these features are described in this chapter and illustrated with examples.

String Variables

EES provides both numerical and string variables types. A string variable holds character string information. A string variable is identified to EES with a variable name that ends with the $ character, as in the BASIC language. The variable name must begin with a letter and consist of 30 or fewer characters, including the $ character.

String variables can be set to string constants. A string constant is a set of up to 255 characters enclosed within single quote marks, e.g.

A$=’carbon dioxide’

String variables can be set equal to other string variables, e.g.,

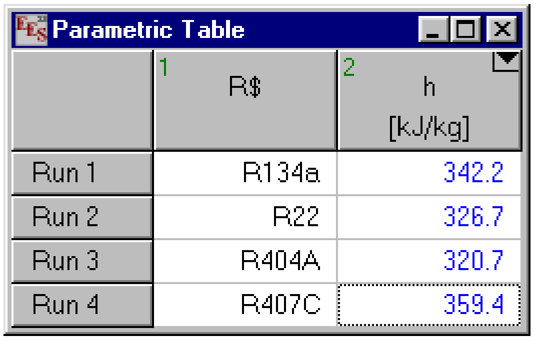
B$=A$

String variables may be passed as arguments to internal functions, procedures and modules or to external functions and procedures as described in Chapter 6.

In general, string variables can be used in EES equations anywhere in which character information is provided. For example, the name of a fluid provided to a thermophysical property function may be a string variable, e.g.,

h=enthalpy(R$,T=T,P=P)

String variables may be used with the Parametric table. In the example below, a Parametric table is used to tabulate the values of specific enthalpy for four refrigerants. Note that the single quote marks which are normally used when specifying a string constant should not be used when entering the strings in the Parametric table.



Parametric Table

	1 R$	2 h [kJ/kg]
Run 1	R134a	342.2
Run 2	R22	326.7
Run 3	R404A	320.7
Run 4	R407C	359.4

A string variable may be used to hold the name of a Lookup file or the name of a column for use with the **Interpolate** or **Lookup** commands, e.g.,

m=Interpolate(File$,Col1$,Col2$,Col1$=x)
k=Lookup(File$,Row,Col$)

String variables may be supplied to the Equations Window using the Diagram Window, either with an edit box or with a pull-down list of alternatives. See the Diagram Window section in Chapter 2 for more information on this capability.

Complex Variables

EES will solve equations involving complex variables of the form a+b*i if the 'Do Complex Algebra' control in the Complex tab of the Preferences dialog is checked. The imaginary number operator may be set to either i or j in the Preferences dialog although i is used in the following discussion.

When set in complex mode, every (non-string) EES variable is represented internally as two variables corresponding to the real and imaginary components of a complex number. The real part is internally identified by appending _r to the variable name. The imaginary part has an appended _i. (You should not use _r or _i at the end of a variable name unless you are specifically referring to the real or imaginary component.) If, for example, you enter the equation

X=Y

EES will automatically create variables X_r, X_i, Y_r and Y_i corresponding to the real and imaginary components of the variables. You will normally not have to refer to the renamed variables, although they will appear with these names in the Variable Info and New Parametric Table dialogs, as well as in column headers of the Parametric Table. However, you can set the value of a real or imaginary part of a complex number by entering the real or imaginary variable name on the left of an equation in the Equations window. For example, the following equation will set the imaginary part of variable omega to 0.

omega_i=0

Any later attempt to set the imaginary part of omega will cause EES to display an error message. If, for example, you also enter the equation, omega = 3 in the Equations window, EES will present an error message when you attempt to solve because the omega=3 equation sets both the real and imaginary parts of omega and the imaginary part would have already been set. You could, of course, enter omega_r=3.

Complex numbers can be entered in either rectangular or polar form. In rectangular form, the complex number is entered using the imaginary number operator (i or j) with a multiplication symbol (*) separating the imaginary number operator from variables or constants. A complex constant can be entered in polar form by entering the magnitude (also called absolute value) of the number and the angle separated with the < symbol. The < character will be displayed in the Solution and Formatted Equations windows as ∠. The angle can be entered in either degrees or radians. If no designation is provided, the angle is assumed to be in the same units as indicated for trigonometric functions in the Unit System dialog. However, you can ensure that the angle you enter is in degrees or radians regardless of the unit system setting by appending deg or rad to the number (with no spaces). For example, the value of Y can be set to the same complex constant in any one of the following three ways.

Y=2 + 3 * i

Y=3.606 < 56.31deg

Y=3.606 <0.9828rad

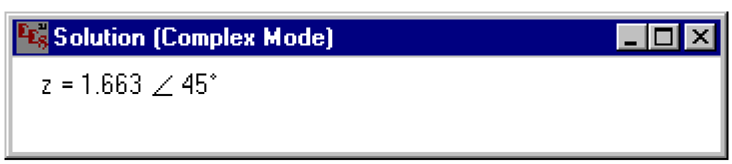
The use of deg or rad to indicate the units of the angle is strongly recommended for three reasons. First, the value of the constant you enter will not be changed by the unit system setting. Second, you can enter complex variables in polar form with the angle in degrees yet do all calculations with the unit setting in radians which is more efficient. Third, if you designate the angle to be in degrees, the degree sign will be shown with the angle in the Formatted Equations window. The output display of the number in the Solution Window can be set for degrees or radians (in polar notation) independent of how the number is entered. The display is changed by clicking the right mouse button on the value and selecting the display options from the Format Variable dialog.

Internally, EES creates two equations for each equation that is entered in the Equations window. One equation is used to equate the real parts of the variables whereas the second equation equates the imaginary parts. The actual set of equations used in complex mode is most clearly seen by viewing the Residuals window which displays the residual and blocking order for each equation. Equations used for the real and imaginary parts are identified with (r) and (i), respectively.

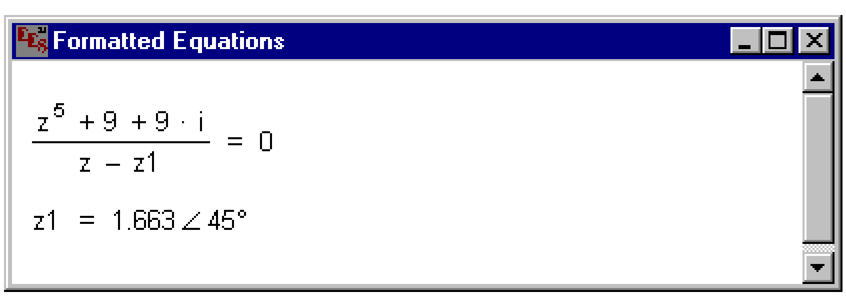
When configured in complex mode, some EES functions, such as **Min** and **Max**, are not accessible. However, most of the built-in functions (including the thermophysical property functions) have been modified to work with complex numbers. For example, the **sin**, **cos**, **ln**, **exp**, and **tanh** functions will accept and return appropriate complex numbers. User-written functions, procedures, and external routines can be used but they will accept and return real numbers only. (Modules are currently not supported in complex mode.) Only the real part of a complex variable will be placed in the argument list of internal or external functions, procedures, and modules.

There are a few built-in functions that operate only in complex number mode. They are **Real**, **Imag**, **Magnitude**, **Angle**, **AngleDeg**, **AngleRad**, and **Conj**. These functions all take one (complex number) argument and set the real part of the complex variable to the selected value.

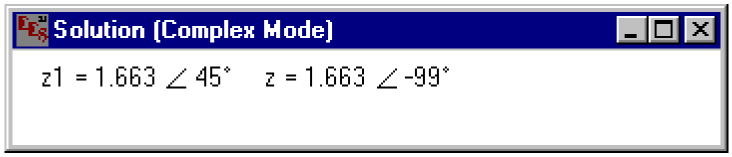
One limitation of the manner in which EES implements complex numbers is that EES can only return one solution, although two or more solutions may exist. There is, however, a simple way to coax EES into providing multiple solutions. Consider the problem of determining the five roots to the complex equation $z^5 + 9 + 9\ i = 0$. Entering this equation into the Equations window and solving with the default guess values will produce a solution of z=1.176+1.176 i. Setting the Solution window display to polar (degree) coordinates will produce the following solution.



This is a correct solution to the equation, but there are four others. To find a different solution (without changing guess values), divide the equation by the difference between z and the value of this root. The Formatted Equations window will appear as shown.

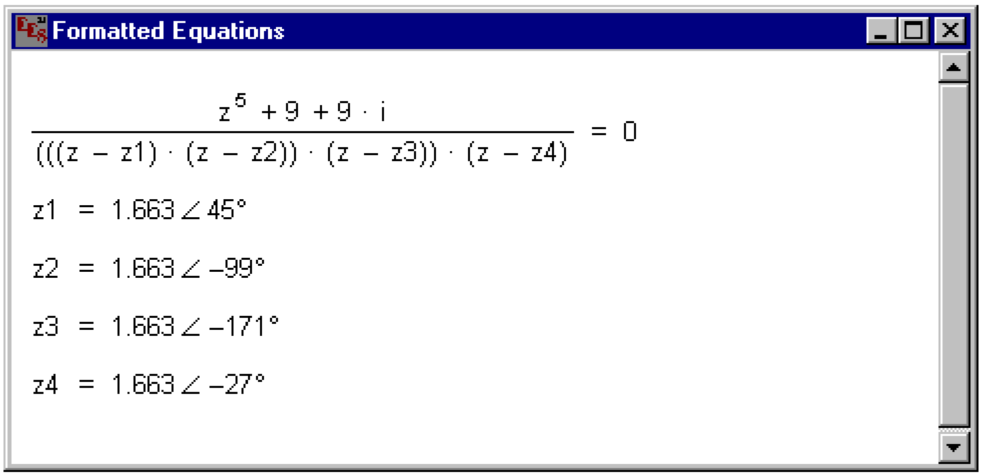


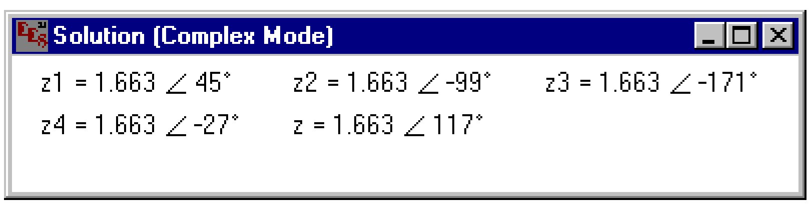
Now, a second solution will be found.



This process can be repeated to find the third, fourth, and fifth roots.

The Formatted Equations window and solution for this last step are shown below.





Array Variables

EES recognizes an array variable by placing the array index within square brackets, e.g. X[5]. Multi-dimensional array variables may also be used with the indices separated by commas, e.g., Z[1,2,3]. The special requirements pertaining to array variables are:

1. An array index may be an integer number, an EES variable which has been previously set to a constant value, the **TableRun#** function, or an algebraic expression involving these quantities with operators +, –, *, and /. Index arithmetic is done left to right with no operator precedence. For example, X[2*3+1] is a valid array variable which EES will transform into X[7]. X[1+2*3] will be transformed into X[9]. The index variable for the DUPLICATE command or the **sum** or **product** functions can also be used in an expression for the array index as shown below.
2. Valid index values range between –32760 and +32760, including zero.
3. The right bracket must be the last character in the variable name.
4. The total length of the variable name, including the brackets and the integer value of the index, must not exceed 30 characters.

EES treats array variables in a very different manner than FORTRAN or Pascal. In EES, each array variable, such as X[99], is a unique variable name. As such, X[99] appears to EES just like any other variable such as ZZZ. The guess value and bounds (along with other information) may be specified for X[99] with the Variable Info command, just as for any other variable. It is legal (but not good practice) to have EES variables names of X, X[1], X[2,3] all within the same equation set. The fact that X[99] appears in the Equations window does not cause EES to reserve memory for 99 elements. Memory is allocated only for the variables which appear in the equations.

Array variables can be useful in several ways. They provide a means of grouping variables of similar type. For example, the temperatures at each state in a system can be written as T[1], T[2], etc. Array variables can be plotted. For example, the temperature and entropy of each state in a thermodynamic cycle can be overlaid on a T-s property diagram. See Property Plot in the Plot Menu section of Chapter 3 for details. Finally, array variables can be used with the DUPLICATE command and the **sum** and **product** functions to provide matrix capability and thereby significantly reduce the amount of typing needed to specify some problems.

The DUPLICATE Command

The DUPLICATE command provides a shorthand way of entering equations into EES. The equations which are to be duplicated are enclosed between the DUPLICATE and END command words. DUPLICATE is useful only when used with array variables. For example, the following statements:

```
N=5
X[1]=1
DUPLICATE j=2,N
    X[j]=X[j–1]+j
END
```

are equivalent to:

```
X[1]=1
X[2]=X[1]+2
X[3]=X[2]+3
X[4]=X[3]+4
X[5]=X[4]+5
```

Note that, within the scope of the DUPLICATE command, the DUPLICATE index variable (j in the example above) can be used in an algebraic expression for the array index. The DUPLICATE index is not an EES variable, but rather just a temporary placeholder for the integers applied in the DUPLICATE command.

The special format requirements pertaining to the DUPLICATE command are as follows:

1. The DUPLICATE command must be on its own line in the Equations window or separated from other equations with a semicolon.

2. The lower and upper limits specified for the index variable in the DUPLICATE command must be integers, EES variables previously assigned to constant values, or the **TableRun#** function.

3. DUPLICATE commands may be nested within one another as deep as desired. However, each DUPLICATE command must use a different index variable name and each must be terminated with an END command. The lower or upper limit of an internal DUPLICATE may be the index value of an external DUPLICATE, e.g.,

   DUPLICATE i=1,5; DUPLICATE j=i,6; X[i,j] = i*j; END; END

4. The END command terminates the last opened DUPLICATE command.

Matrix Capabilities

Many engineering problems can be formulated into a linear system of algebraic equations of the form

$$[\mathbf{A}]\ [X] = [B]$$

where [**A**] is a square matrix of coefficients, and [X] and [B] are vectors. Ordinarily, the matrix equation is solved to determine the elements in the vector [X] for known [**A**] and [B]. In this case,

$$[X] = [\mathbf{A}]^{-1}\ [B]$$

EES can directly solve the equations represented by [**A**] [X] = [B] by entering each equation directly into the Equations window in any format or order. However, a more elegant and convenient method for solving these equations in EES is to make use of the matrix capability. EES can solve matrix equations, formulated with array variables, by using the DUPLICATE command and the **sum** function. For example, consider the following radiation heat transfer problem in which [A] and [B] are given below, and the radiosity vector, [X] is to be determined.[14]

$$[A] = \begin{bmatrix} 10 & -1 & -1 \\ -1 & 3.33 & -1 \\ -1 & -1 & 2 \end{bmatrix} \qquad [B] = \begin{bmatrix} 940584 \\ 4725 \\ 0 \end{bmatrix}$$

The equations required in EES to solve this problem are as follows:

```
Equations Window: C:\EES32\examples\Matrix2.ees
{Define the A matrix}
A[1,1]=10;     A[1,2]=-1;     A[1,3]=-1
A[2,1]=-1;     A[2,2]=3.33;   A[2,3]=-1
A[3,1]=-1;     A[3,2]=-1;     A[3,3]=2

{Define the B vector}
B[1]=940584;     B[2]=4725;     B[3]=0

{Now let B=A*X}
DUPLICATE i=1,3
     B[i]=sum(A[i,k]*X[k],k=1,3)
END
```

The calculated elements in the X array will appear in the Arrays window.

---

[14] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 13

Arrays Table

	1 $A_{i,1}$	2 $A_{i,2}$	3 $A_{i,3}$	4 B_i	5 X_i
[1]	10.00	-1.00	-1.00	940584	108339
[2]	-1.00	3.33	-1.00	4725	59093
[3]	-1.00	-1.00	2.00	0	83716

Note that it was not necessary to determine the inverse of [**A**] to obtain the solution. EES calculates the inverse matrix internally, as needed to solve these and any other simultaneous equations. However, the inverse matrix $[\mathbf{A}]^{-1}$ can be determined by setting the matrix product [**A**] $[\mathbf{A}]^{-1}$ equal to the identity matrix in the following manner.

Equations Window: C:\EES32\examples\Matrix2.ees

```
{Set up identity matrix using Step function}
N=3
DUPLICATE i=1,N
    DUPLICATE j=1,N
        Identity[i,j]=1-step(abs(i-j)-1)
    END
END
{Set identity matrix to the product of A and Ainv}
DUPLICATE i=1,N
    DUPLICATE j=1,N
        Identity[i,j]=sum(A[i,k]*Ainv[k,j],k=1,N)
    END
END
```

The inverse matrix Ainv will appear in columns of the Arrays window.

Arrays Table

	4 $Ainv_{i,1}$	5 $Ainv_{i,2}$	6 $Ainv_{i,3}$
[1]	0.11	0.06	0.09
[2]	0.06	0.39	0.22
[3]	0.09	0.22	0.66

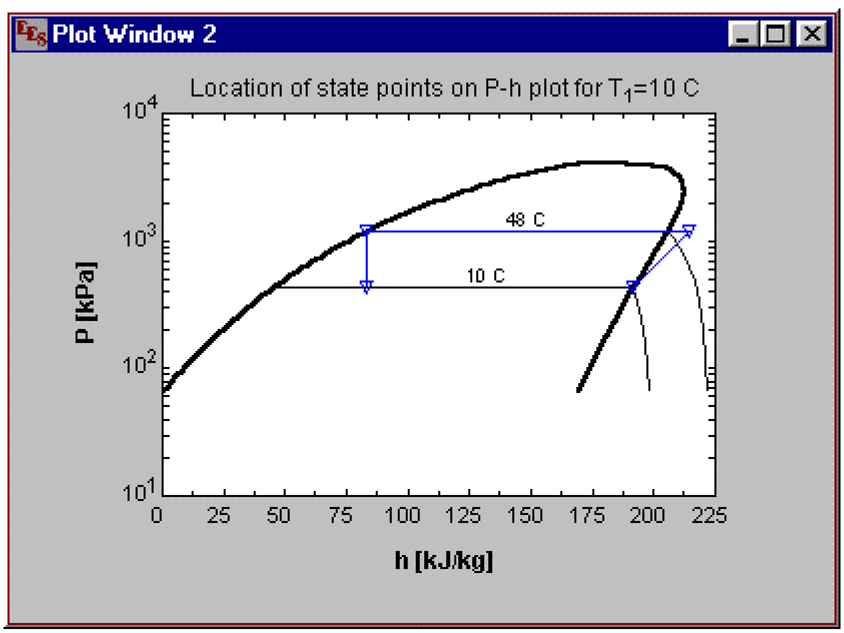
The two examples above provide a general procedure for determining the product of a matrix and a vector or the product of two matrices. Using the DUPLICATE command along with array variables in EES is no more efficient than the alternative of entering each equation separately with non-array variable names; however, the matrix capabilities in EES can significantly reduce the amount of typing required to enter the problem and, more importantly, make the equations easier to follow.

Using the Property Plot

The Property Plot menu item in the Plot menu generates T-s, T-v, P-v, or P-h diagrams for any of the fluids in the EES data base. A psychrometric chart is generated if substance AirH2O is selected. The property plot is placed in one of the EES plot windows.

Additional property data or thermodynamic state point information can be superimposed on the property plots using the Overlay Plot command in the Plot menu. This is most conveniently done if array variables are used for the thermodynamic variables. Another benefit of using array variables is that the state property data then appear in the Arrays Table in a convenient tabular form.

The P-h plot below shows the state points for simple refrigeration cycle operating between an evaporator temperature of 10°C and a condensing temperature of 48°C with a compressor isentropic efficiency of 0.70. The plot was prepared by first producing a P-h plot for R12 with isotherms at 10°C and 48°C with the Property Plot command and then overlaying the P[i] and h[i] arrays (plotting from the Arrays table) for the four state points in a refrigeration cycle analysis. The equations can be found in file REFRIG.EES in the Examples subdirectory.



Integration and Differential Equations

The INTEGRAL function is used to evaluate an integral and in the solution of differential equations. The format of the **Integral** function is:

$$\int_{t1}^{t2} f \, dt = \textbf{Integral}(f,t)$$

There are two basic forms of the integral function which differ in their reliance on the Parametric table.

<u>Table-based Integral function</u>

The table-based Integral function uses the Parametric table to provide the limits and step size of the integration variable. The format of this function is **Integral**(f, t). This form of the Integral function can be used only in conjunction with the Parametric Table. The integration variable, t, must be a legal variable name which has values defined in one of the columns of the Parametric table. The limits on the integration variable, t_1 and t_2 are the first and last values specified for variable t. The integrand, f, can be a variable or any implicit or explicit algebraic expression involving variables, values, and the integration variable, t.

<u>Equation-based Integral function</u>

The Equation-based integral function serves the same purpose as the table-based integral function but it does not require the use of the Parametric table. The format for the equation-based integral function is

F = INTEGRAL(f, t, t1, t2, tStep)

or

F = INTEGRAL(f, t, t1, t2) {automatic step size}

t1 and t2 are the lower and upper limits of the integration variable. These limits may be specified with a constant or an EES expression. However, the limits cannot be a function of the integration variable t or any other variable which changes during the course of the integration.

tStep is the increment EES will use for the integration variable while numerically evaluating the integral between the specified limits. tStep cannot vary during the course of the integration. Note that specification of tStep is optional. If tStep is not provided, EES will select the stepsize using an automatic stepsize adjustment algorithm.

EES uses a second-order predictor-corrector algorithm for evaluating the integral. This algorithm is designed for solving combined algebraic and differential equations which result when the integrand is a complex function of other variables. The algorithm is especially suited to stiff equations.

EES uses the **Integral** function to solve initial value differential equations. Any first-order differential equation can be transformed into an appropriate form by integrating both sides. For example, the differential equation, dy/dx = f(x,y) can be equivalently written as

$$y = y_0 + \int f(x,y)\,dx$$

where y_0 is the initial value of y. This equation can be solved using either the table-based or equation-based forms of the **Integral** function. The table-based form would be entered into the EES Equations Window as

y = y0 + INTEGRAL(fxy, x)

where fxy is an EES variable or expression. To solve the equation it is necessary to create a Parametric table containing a column for variable x. Values of x are entered into the Parametric table with the value in first row corresponding to the lower limit of x and the value in the last row corresponding to the upper limit. The step size is determined by the difference between the value of x in successive rows and need not be a fixed value. The integral is evaluated when the **Solve Table** command is applied.

The equation-based form would appear in the EES Equations window as

$y = y_0$ + INTEGRAL(fxy, x, low, high)

y_0 and fxy are as defined above. Low and high are the lower and upper limits for x. A Parametric table is not required for use with this form of the **Integral** command. Since the stepsize is not specified, EES will use automatic stepsize selection.

Solving First-Order Initial Value Differential Equations

Initial-value differential equations can be solved in a number of ways with EES. Chapter 5 describes a Library function included with EES in the USERLIB subdirectory which implements a 4^{th} order Runge-Kutta algorithm. This method can only be used if the derivative can be expressed explicitly as a function of the dependent and independent variables. This section demonstrates two ways of solving simultaneous algebraic and differential equations using the **Integral** function or the **TableValue** function in conjunction with the Parametric Table.

Method 1: Solving Differential Equations with the Integral Function

Consider the problem of determining the time–temperature history of a sphere of radius r=5 mm, initially at a uniform temperature of 400°C. The sphere is exposed to 20°C air with a convection coefficient of h=10 W/m^2-K. The thermophysical properties of the sphere material are:

ρ = density = 3000 kg/m^3
k = thermal conductivity = 20 W/m-K
c = specific heat = 1000 J/kg-K

Calculation of the Biot number will indicate that the sphere can be treated as a lumped system, and therefore it may be assumed to be at a uniform temperature at any instant of time.[15] The relation between the sphere temperature and time is given by an energy balance on the sphere, which results in the following differential equation

$$-h\,A\,(T - T_\infty) = \rho\,c\,V\,\frac{dT}{dt}$$

where

h	is the convective heat transfer coefficient
T	is the uniform temperature of the sphere at any time
T_∞	is the temperature of the air stream = 20°C
A	is the surface area of the sphere = 4 π r^2
V	is the volume of the sphere = 4/3 π r^3
t	is time

This differential equation has the following analytic solution which can be used to check the accuracy of the numerical solution provided by EES.

$$\frac{T - T_\infty}{T_i - T_\infty} = \exp\left(\frac{-h\,A}{\rho\,c\,V}\,t\right)$$

To solve the differential equation numerically in EES, enter the following equations.

```
Equations Window: C:\EES32\examples\Difeqn1.ees
"Physical properties"
r=0.005; A=4*pi*r^2; V=4/3*pi*r^3
"Material properties"
rho=3000; c=1000
"Constants"
Tinf=20; Ti=400; h=10
"Energy balance to determine dTdt"
rho*V*c*dTdt=-h*A*(T-Tinf)

"Integrate dTdt to find T as a function of time"
T=Ti+integral(dTdt,Time)
"Exact solution"
(Texact-Tinf)/(Ti-Tinf)=exp(-h*A/(rho*c*V)*Time)
```

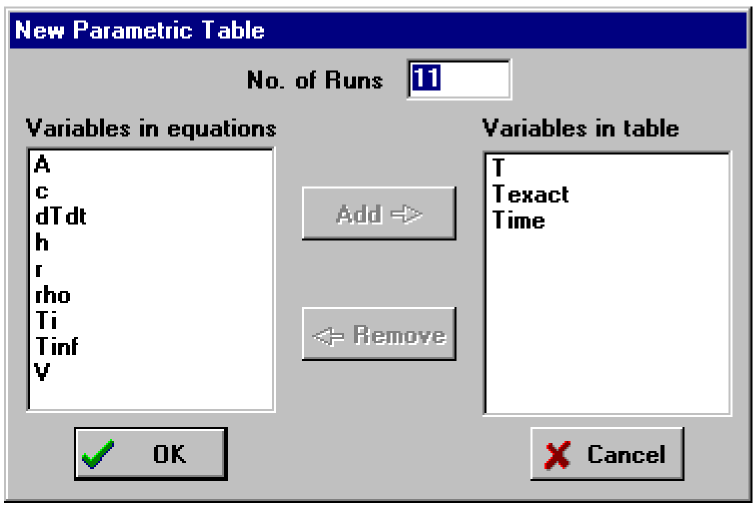
Next, a Parametric Table is generated with the New Table command in the Parametrics menu. (Note that, if the start and stop times were provided as the third and fourth parameters for the

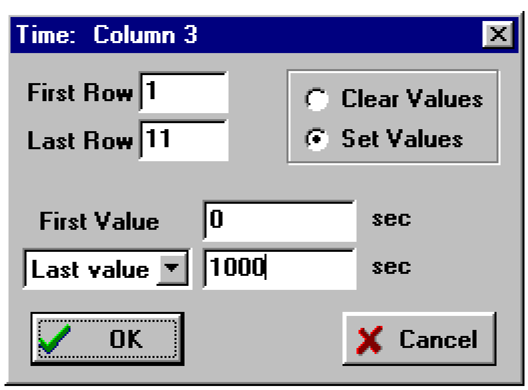
[15] Incropera, F.P. and DeWitt, D.P., *Fundamentals of Heat and Mass Transfer*, 2nd edition, John Wiley and Sons, 1985, Chapter 5

Integral function, it would not be necessary to use the Parametric table. However, intermediate results for times between the start and stop times would not be available for plotting.)

Select T, Time, and Texact as the three variables to include in the table. Enter 11 runs which will allow a time–temperature history for 1000 seconds starting at 0 with 100 second intervals. The New Table dialog window should now appear as shown below.

Click the OK button. It is next necessary to enter the values of Time in the table for which the temperature T is to be calculated. A timestep of 100 seconds has been chosen. With a fixed timestep, the values of Time can be most easily entered by clicking on the ▼ control at the upper right of the Time column header cell. Enter 0 for the First Value. Set the drop-down list control to 'Last Value' and enter 1000 for the last value as shown.

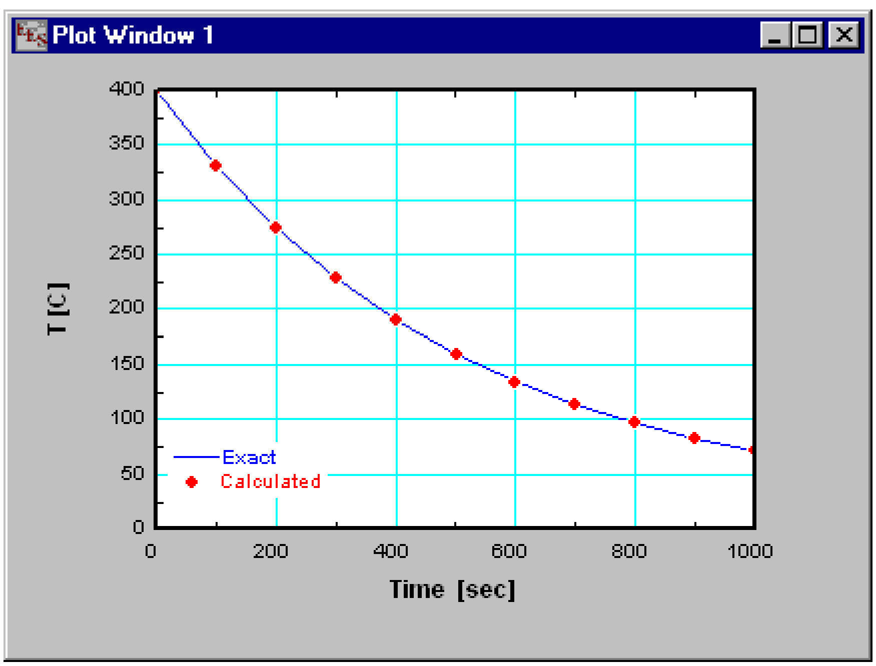




The value of Time from 0 to 1000 will be automatically entered into the table when you click the OK button and displayed in normal type. Now select Solve Table from the Calculate menu to calculate the numerical and analytical values of temperature corresponding to each value of Time in the table. When the calculations are completed, the Parametric Table window will display the solutions. Calculated values are shown in bold. (The format of calculated values in the Parametric Table can be set using the Preferences command in the Options menu.) The plot shows that the numerically determined temperature agrees closely with the exact analytic solution.

Parametric Table

	1 T [C]	2 Texact [C]	3 Time [sec]
Run 1	400.00	400.00	0
Run 2	330.91	331.12	100
Run 3	274.38	274.72	200
Run 4	228.13	228.55	300
Run 5	190.29	190.75	400
Run 6	159.33	159.79	500
Run 7	133.99	134.45	600
Run 8	113.27	113.71	700
Run 9	96.31	96.72	800
Run 10	82.44	82.81	900
Run 11	71.08	71.43	1000



Method 2: Solving Differential Equations with the TableValue Function

In this section, we solve the same first-order differential equation described in Method 1.

$$-h\,A\,(T - T_\infty) = \rho\,c\,V\,\frac{dT}{dt}$$

The differential is approximated as

$$\frac{dT}{dt} \simeq \frac{T^{new} - T^{old}}{\Delta t}$$

T^{new} is the temperature at the current time which is to be calculated. T^{old} is the temperature at the previous time which can be found in the previous row of the Parametric table using the **TableValue** function. The **TableValue** function returns the value in the Parametric Table at a specified row and column, as described in Chapter 4. With this function, it is possible to access the values of variables calculated in previous runs during the Solve Table calculations. Δt is the timestep which, in the equations shown below, is the difference of the current and previous values of the variable Time.

Both an explicit method (Euler's method) and an implicit method (Crank-Nicolson) are used to solve this first-order differential equation and compared with the exact solution. In the Euler method, only previous temperatures are used to evaluate the right-hand side of the differential equation. In the Crank-Nicolson method, the average of the previous and current temperatures is used. The Crank-Nicolson method is implicit because the current temperature is not as yet determined. The implicit method is no more difficult to implement since EES is designed to solve implicit equations. Shown below is a listing of all of the equations needed to solve this problem.

Most of the equations are identical to those used for Method 1. T_Euler is the temperature calculated by Euler's method. T_CN is the temperature calculated by the Crank-Nicolson method. (In the Formatted Equations and Solution windows, these variables will display as T_{Euler} and T_{CN}, respectively.) To proceed, a Parametric table must be defined, as in Method 1. The values of T_Euler, T_CN and T_exact in the first row of the table, corresponding to Time=0, are the initial conditions and their values (400°C) must be entered. Then, the Solve Table command is used to complete the table, with calculations starting at Run 2. Note the use of the TableRun# function to return the row number in the Parametric Table for which calculations are currently being done, and the TableValue function which returns the value at a specified row and column in the Parametric Table.

Equations Window: C:\EES32\examples\Difeqn2.ees

```
"Physical properties"
r=0.005; A=4*pi*r^2; V=4/3*pi*r^3
"Material properties"
rho=3000; c=1000
"Constants"
Tinf=20 {C}; Ti=400 {C}; h=10 {W/m2-K}; delta=100 {s}
"Finite difference energy balance"
"Euler Method"
T_Euler_old=tablevalue(TableRun#-1,#T_Euler) "retrieves previous T_Euler"
rho*V*c*(T_Euler-T_Euler_old)/delta=-h*A*(T_Euler_old-Tinf)
"Crank-Nicolson Method"
T_CN_old=tablevalue(TableRun#-1,#T_CN) "retrieves previous T_CN"
rho*V*c*(T_CN-T_CN_old)/delta=-h*A*((T_CN_old+T_CN)/2-Tinf)
"Exact solution"
(T_exact-Tinf)/(Ti-Tinf)=exp(-h*A/(rho*c*V)*Time)
```

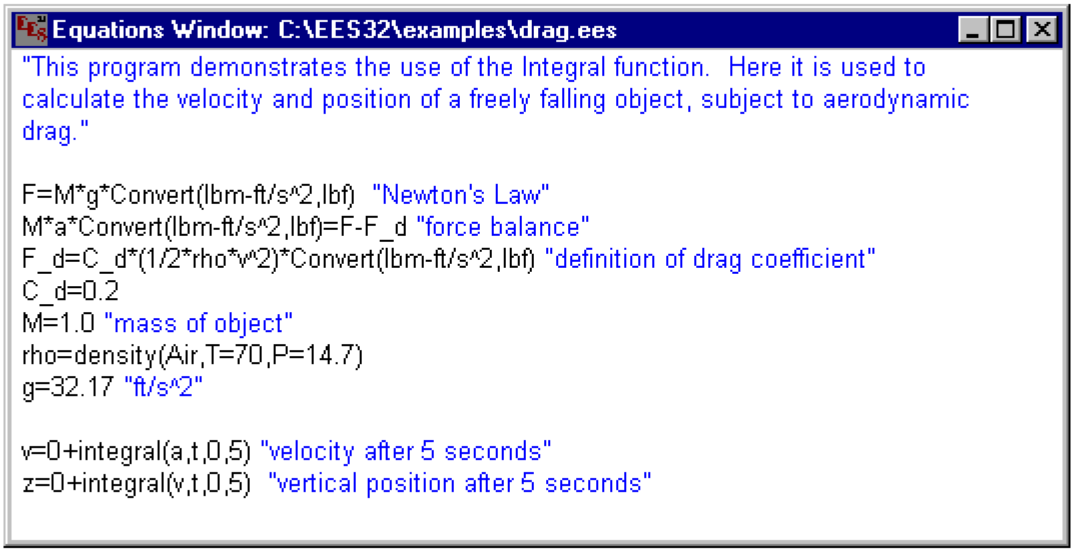
Shown below is the completed table with the numerical and analytical solutions. Calculated values are shown in italics. It is evident that Euler's method does not provide as accurate a solution as that obtained with the **Integral** function (Method 1) or the Crank-Nicolson method. Improved accuracy could be obtained by reducing the time step, but this would require additional computational effort and storage space (which is not significant here).

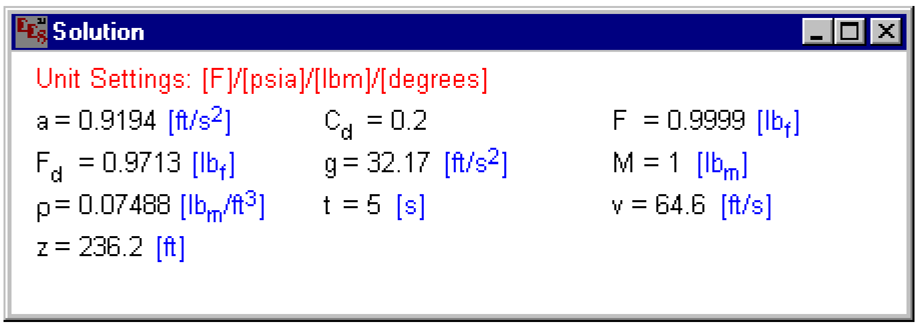
Parametric Table

	1 Time [sec]	2 T_{Euler} [C]	3 T_{CN} [C]	4 T_{exact} [C]
Run 1	0	400.0	400.0	400.0
Run 2	100	*324.0*	*330.9*	*331.1*
Run 3	200	*263.2*	*274.4*	*274.7*
Run 4	300	*214.6*	*228.1*	*228.5*
Run 5	400	*175.6*	*190.3*	*190.7*
Run 6	500	*144.5*	*159.3*	*159.8*
Run 7	600	*119.6*	*134.0*	*134.5*
Run 8	700	*99.7*	*113.3*	*113.7*
Run 9	800	*83.8*	*96.3*	*96.7*
Run 10	900	*71.0*	*82.4*	*82.8*
Run 11	1000	*60.8*	*71.1*	*71.4*

<u>Solving Second and Higher Order Differential Equations</u>

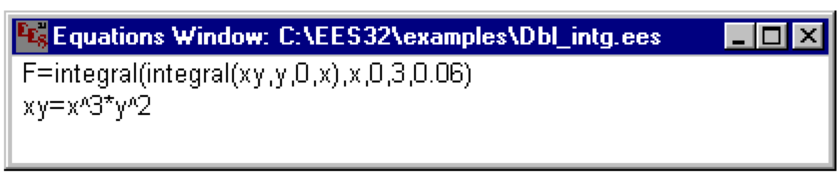
Higher order differential equations can also be solved by repeated use of the **Integral** function. Shown below is an EES program which solves a second-order differential equation to calculate the velocity and position of a freely falling object, subject to aerodynamic drag. The Solution Window appears after the Solve command (F2) is issued.

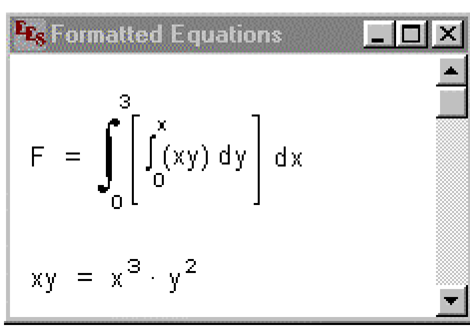


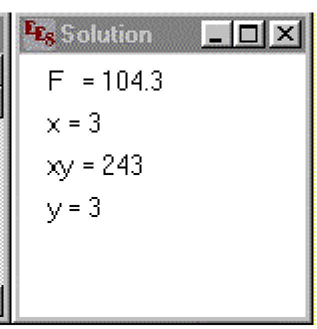


<u>Multiple-Variable Integration</u>

Multiple integration is provided by nesting calls to the **Integral** function. Up to six levels can be nested. The following example performs a numerical double integration.







<u>Combined Use of the Table and Equation-Based Integral Functions</u>

An advantage of the table-based form of the **Integral** function over the Equation-based form is that intermediate values are provided in the Parametric table so that the trajectory of the integrated quantity may be observed and plotted. A possible disadvantage is that the integrated variable must appear in the table for each step of the integration. It is possible to combine the equation-based form of the **Integral** function with the Parametric table so that only some of the intermediate results are reported in the Parametric table, as demonstrated in the following example.

Equations Window: C:\EES32\examples\Substeps.ees

```
"This example illustrates how EES  can solve a differential equation using a small integration
step size while reporting integrated values at larger specified intervals. This process is illustrated
by solving the following 2nd order differential equation:
    y``+y`-2y=4x  with y`(0)=0 and y(0)=1.
The Parametric table is used to tabulate calculated values of y and y` at desired values of x.
The initial values of y and y` are provided in the first row of the table.  Calculations start with the
second row.  The Integral functions below integrate y`` and y` for values of x ranging between the
values in the preceding and current  rows of the table.  Thus, calculations for Run 2 integrate
over values of x  between 0 and 0.1 while calculations for Run 3 integrate for values of x between 0.1
and 0.2.  Automatic step size is used in the integration.  The net result is that many steps are used
to evaluate the integral, but values are only reported in the Parametric table for increments of 0.1 of x.
"
y``=4*xx+2*y-y`
y`=y`_i+integral(y``,xx,x_i,x)
y=y_i+integral(y`,xx,x_i,x)
y_i=tablevalue(Row-1,#y)
y`_i=tablevalue(Row-1,#y`)
x_i=tablevalue(Row-1,#x)
```

Parametric Table

	1 Row	2 x	3 y	4 y`
Run 1	1	0	1.0000	0
Run 2	2	0.1	1.0103	0.2103
Run 3	3	0.2	1.0428	0.4428
Run 4	4	0.3	1.0997	0.6997
Run 5	5	0.4	1.1837	0.9837
Run 6	6	0.5	1.2974	1.297
Run 7	7	0.6	1.4442	1.644
Run 8	8	0.7	1.6275	2.028
Run 9	9	0.8	1.8511	2.451
Run 10	10	0.9	2.1192	2.919
Run 11	11	1	2.4366	3.437

Appendix A

Hints for Using EES

1. The Variable Information command in the Options menu produces an alphabetical list of all variables appearing in the Equations window. Check this list to make sure that you have not misspelled a variable name.

2. The Residuals window provides an indication of the accuracy in which each non-trivial equation in the Equations window was solved *and* the order in which the equations are solved. An examination of the residuals indicates which equations were not solved when EES indicates that a solution could not be found.

3. If your equations do not converge, it may be that the guess values are poor. In this case, the problem can often be solved by entering equations which set guess values for one or more of the unknown variables and modifying the equations as needed to ensure an equal number of variables and equations. If a solution is then obtained, use Update Guesses in the Calculate menu to set the guess values of all variables to their current values. Then return the Equations window to its original form and solve again.

4. If EES is unable to solve your set of nonlinear equations, try exchanging some independent and dependent variables to produce an equation set which is easier to solve. For example, EES may not be able to solve the following heat exchanger equations to determine NTU with the default guess values and bounds.

   ```
   Eff=.9
   Cmax = 432
   Cmin = 251
   eff = (1 - exp(-NTU * (1 - (Cmin/Cmax)))) / (1 - (Cmin/Cmax) * exp(-NTU *
       (1 - (Cmin/Cmax))))
   ```

   However, the equations would be easily solved if the value of NTU were specified in place of Eff.

   ```
   NTU = 5
   Cmax = 432
   Cmin = 251
   eff = (1 - exp(-NTU * (1 - (Cmin/Cmax)))) / (1 - (Cmin/Cmax) * exp(-NTU *
       (1 - (Cmin/Cmax))))
   ```

   A few trials will indicate that NTU must be between 3 and 5 for Eff = 0.9. Setting a guess value for NTU of 4 allows EES to quickly determine the final value of 3.729.

5. A sure way to solve difficult problems with EES is to add an additional variable so that the problem has one more degree of freedom. Then, use the Parametric Table to vary the values of one of the implicit variables in order to find the solution in which the additional variable has a value of zero. For example, consider the following radiation calculation in which the value of T is to be determined. The first three equations must be solved simultaneously and they are non-linear because T is raised to the fourth power. EES may have trouble determining the solution, depending upon the guess values.

   ```
   QL = AL*Sigma*(T^4 - TL^4)
   QB = AH*Sigma*(TH^4 - T^4)
   QL = QB
   Sigma=0.1718E-8
   AL=.5;  AH=1;  TL=300;  TH=1000
   ```

   Alternatively, add a variable, Delta, such that

   ```
   QL = AL*Sigma*(T^4 - TL^4)
   QB = AH*Sigma*(TH^4 - T^4)+Delta
   QL = QB
   Sigma=0.1718E-8
   AL=.5;  AH=1;  TL=300;  TH=1000
   ```

   Now, set up a Parametric Table containing variables T and Delta. Use the Alter Values command to set a range of values of T and Solve Table to calculate the corresponding values of Delta. The value(s) of T for which Delta is zero constitute a solution to the equation set. The New Plot Window command is handy for visualizing the relationship between T and Delta. If the values of Delta do not cross zero, there is no solution to the equation set for the range of T values investigated. This is perhaps the most useful method of solving a difficult set of non-linear equations.

6. The Store button in the Default Info dialog can be particularly convenient if you have a customary set of nomenclature for your variable names. For example, if variables beginning with letter T often designate temperatures, set the bounds, display format and units for letter T and then Store the default information. You can later use the Load button to restore this set of default variable information. EES will then always set this information for you in following problems.

7. The arrow keys help some users move about more quickly in the Equations, Parametric and Lookup Tables. In the Equations window, the up and down arrows move the cursor up and down one line; left and right arrow move the cursor left and right one character. Home and End move to the start and end of the current line. In the tables, the arrow keys move to the next cell in the direction of the arrow. The Return and Tab keys produce the same effects as the down arrow and right arrow keys, respectively.

8. Use the Tab key in the Equations window to set off the equations for improved readability.

9. Except for the Steam_NBS substance, the EES property correlations are not specifically applicable in the compressed (sub-cooled) liquid range. Instead, it is assumed that subcooled liquid is incompressible and properties are taken to be those for saturated liquid. Thus, in the subcooled region, $v(T,P)=v(T,P_{sat})$, $u(T,P)=u(T,P_{sat})$ and $s(T,P)=s(T,P_{sat})$. To calculate the ideal work of a pump, for example, recall that h2-h1 = -Wpump = $\int v\, dP$ = v*(P2-P1), since for an incompressible substance, v is independent of P.

10. The Arrays window can be quite useful for organizing the property information in a thermodynamics problem having multiple states. Use array variables, such as T[1], P[1], and h[1] (rather than T1, P1, and h1) for the properties at each state. The state properties will appear in a neat table in the Arrays window, rather than jumbled together in the Solutions window. Be sure the Use Arrays window option in the Display Options dialog has been selected.

11. Considerable effort has been expended to design EES so that it does not unexpectedly quit under any circumstances. However, this may still happen. In this case, EES will save your work in a file called EESERROR before it terminates. You can restart EES and load the EESERROR file so none of your work is lost.

12. Use the $INCLUDE directive to load commonly used constants, unit conversions, or other equations into the Equations window. You won't see them, but they're there, available for use. You can also load library files with the $INCLUDE directive.

13. If you write an EES Library Function which calls any of the built-in thermodynamic or trigonometric functions, use the UnitSystem command to determine the current unit system settings. Then you can use If Then Else statements to ensure that the arguments provided to the thermodynamic or trigonometric functions have the correct values.

14. The background color option for columns in the Parametric Table is useful if you are preparing an EES program in which others will be entering data into the columns, as in a spreadsheet. Set the background color for the columns in which data are to be entered, to distinquish them from the columns in which calculated results will appear.

15. If you are working in Complex Mode, use the $COMPLEX ON directive at the top of the Equations window. Its more convenient than changing the Complex Mode setting in the Preferences dialog.

Appendix B

Numerical Methods Used in EES

EES uses a variant of Newton's method [1-4] to solve systems of non-linear algebraic equations. The Jacobian matrix needed in Newton's method is evaluated numerically at each iteration. Sparse matrix techniques [5-7] are employed to improve calculation efficiency and permit rather large problems to be solved in the limited memory of a microcomputer. The efficiency and convergence properties of the solution method are further improved by the step-size alteration and implementation of the Tarjan [8] blocking algorithm which breaks the problems into a number of smaller problems which are easier to solve. Several algorithms are implemented to determine the minimum or maximum value of a specified variable [9,10]. Presented below is a summary of these methods, intended to provide users with a better understanding of the processes EES uses in obtaining its solutions.

Solution to Algebraic Equations

Consider the following equation in one unknown:

$$x^3 - 3.5\,x^2 + 2\,x = 10$$

To apply Newton's method to the solution of this equation, it is best to rewrite the equation in terms of a residual, ε, where

$$\varepsilon = x^3 - 3.5\,x^2 + 2\,x - 10$$

The function described by this equation is shown in Figure 1. There is only one real solution (i.e., value of x for which $\varepsilon = 0$) in the range illustrated at x = 3.69193.

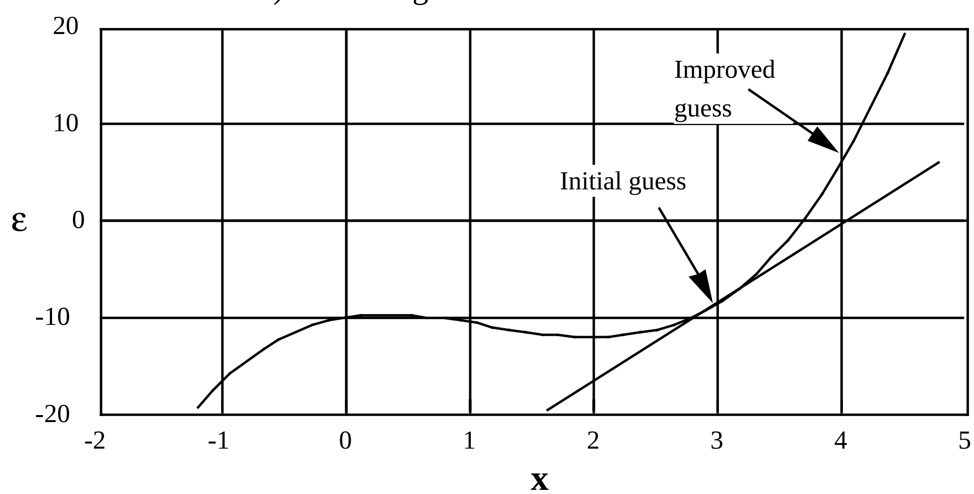


Figure 1: Residual of $x^3 - 3.5\,x^2 + 2\,x = 10$ as a function of x

Newton's method requires an estimate of the total derivative of the residual, J. For this equation, the derivative is:

$$J = \frac{d\varepsilon}{dx} = 3\,x^2 - 7\,x + 2$$

To solve the equation, Newton's method proceeds as follows:

1. An initial guess is made for x (e.g., 3).
2. The value of ε is evaluated using the guess value for x. With x = 3, ε = -8.5.
3. The derivative J is evaluated. With x = 3, J = 8.
4. The change to the guess value for x, i.e., Δx, is calculated by solving J Δx = ε. In this example, Δx is -1.0625.
5. A (usually) better value for x is then obtained as x - Δx. In the example, the improved value for x is 4.0625 (which results in ε = 7.4084).

Steps 2 to 5 are repeated until the absolute value of ε or Δx becomes smaller than the specified tolerances in the Stop Criteria dialog. The method, when it converges, converges quite quickly. However, a bad initial guess can cause the method to diverge or to converge quite slowly. Try, for example, an initial guess of 2 and see what happens.

Newton's method can be extended for solving simultaneous non-linear equations. In this case, the concept of "derivative" generalizes into the concept of "Jacobian matrix." Consider the following two simultaneous equations in two unknowns:

$$x_1^2 + x_2^2 - 18 = 0$$

$$x_1 - x_2 = 0$$

The equations can be rewritten in terms of residuals ε_1 and ε_2:

$$\varepsilon_1 = x_1^2 + x_2^2 - 18 = 0$$

$$\varepsilon_2 = x_1 - x_2 = 0$$

The Jacobian for this matrix is a 2 by 2 matrix. The first row contains the derivatives of the first equation with respect to each variable. In the example above, the derivative of ε_1 with respect to x_2 is 2 x_2. The Jacobian matrix for this example is:

$$J = \begin{bmatrix} 2\cdot x_1 & 2\cdot x_2 \\ 1 & -1 \end{bmatrix}$$

Newton's method as stated above applies to both linear and nonlinear sets of equations. If the equations are linear, convergence is assured in one iteration, even if a "wrong" initial guess was made. Non-linear equations require iterative calculations. Consider the following initial guess:

$$x = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

The values of ε and J for this initial guess are: $\varepsilon = \begin{bmatrix} -10 \\ 0 \end{bmatrix}$ $J = \begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix}$

Improved values for the x vector are obtained by solving the following matrix problem involving the Jacobian and the residual vector.

$$\begin{bmatrix} 4 & 4 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -10 \\ 0 \end{bmatrix}$$

Solving this linear equation results in:

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} -1.25 \\ -1.25 \end{bmatrix}$$

Improved estimates of the x_1 and x_2 are obtained by subtracting Δx_1 and Δx_2, respectively, from the guess values.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3.25 \\ 3.25 \end{bmatrix}$$

The correct solution to this problem is $x_1 = x_2 = 3.0$. The calculated values of x_1 and x_2 are closer to the correct solution than were the guess values. The calculations are now repeated using the most recently calculated values of x_1 and x_2 as the guess values. This process is repeated until convergence is obtained.

The Jacobian matrix plays a key role in the solution of algebraic equations. The Jacobian matrix can be obtained symbolically or numerically. Symbolic evaluation of the Jacobian is more accurate, but requires more processing. Accuracy of the Jacobian, however, does not necessarily lead to more accuracy in the solution, only to (sometimes) fewer iterations. EES evaluates the Jacobian numerically. Because EES does all calculations with 96 bit precision (about 20 decimal places), numeric evaluation of the Jacobian rarely results in convergence problems from loss of precision.

In most equation sets, many of the elements of the Jacobian matrix are zero. A matrix with many zero elements is called a sparse matrix. Special ordering and processing techniques make handling of sparse matrices quite efficient. In fact, without sparse matrix techniques the number of simultaneous equations which could be solved by EES would be substantially less than 6000, the current limit implemented in EES. Further references on sparsity and on how to handle sparse matrices are available in [5, 6]. A collection of routines that are designed to handle very large sparse matrices are described in [7].

Newton's method does not always work, particularly if a "bad" initial guess for the x vector is supplied. The solution obtained after applying the correction Δx to the previous x vector should be more correct (i.e., result in a smaller maximum residual) than the solution obtained before the correction. EES always checks for this condition. If this is not true, EES will halve the step Δx and evaluate the residuals again. If this does not improve the solution, the step is halved again (up to 20 times). If the resultant solution is still not better than the solution prior to the correction, EES will reevaluate the Jacobian and try again until one of the stopping criteria forces the calculations to stop. Step halving is very helpful when a bad initial guess is provided. Figure 2 illustrates the process for the solution of the single equation in the first example, starting from a guess of x=2.5. In this case, step halving works quite well.

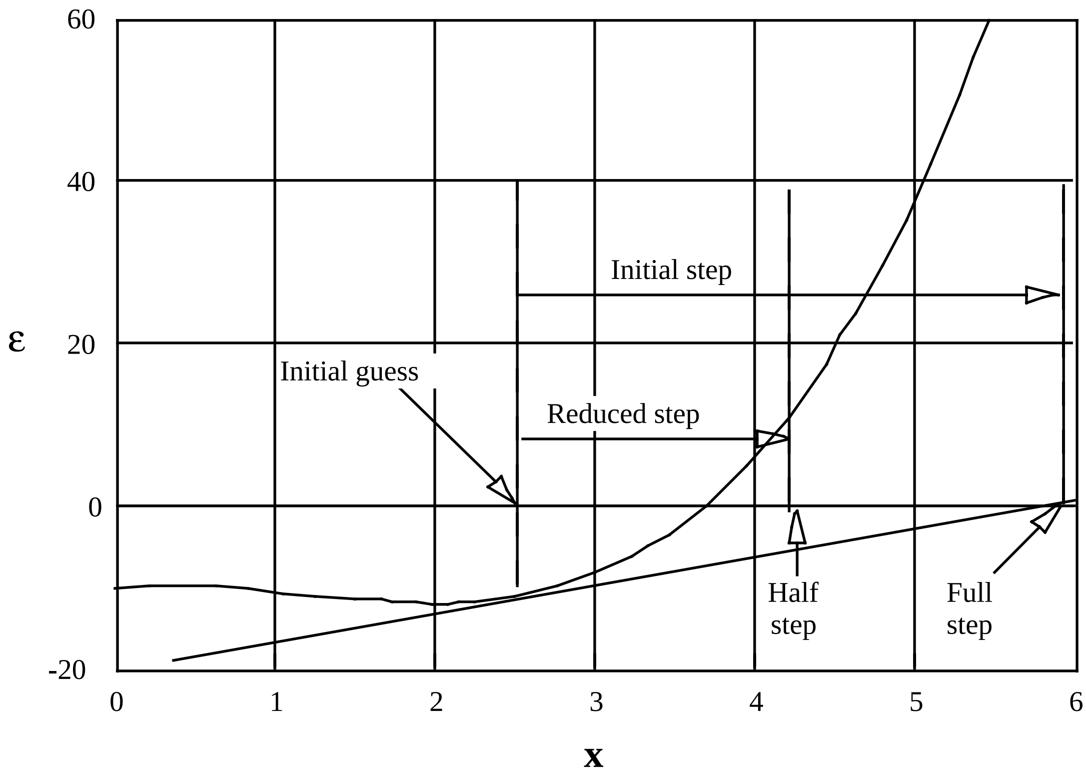


Figure 2: Use step-halving for improving the convergence

Blocking Equation Sets

Even though you may have what looks like a set of simultaneous equations, it is often possible to solve these equations in groups (sometimes one at the time) rather than all together as one set. Solving equations in groups makes Newton's method work more reliably. For this reason, EES organizes the equations into groups (or blocks) before solving. Consider, for example, the following set of equations:

$$x_1 + 2\,x_2 + 3\,x_3 = 11$$
$$5\,x_3 = 10$$
$$3\,x_2 + 2\,x_3 = 7$$

These equations can be solved as one simultaneous set. However, they can be more easily solved if they are reordered and blocked. It is better to re-order first. EES automatically recognizes that equation 2 can be solved directly for x_3. Once this is done, equation 3 can be solved for x_2. Finally, equation 1 can be solved for x_1. This results in three blocks of equations, each with one equation and one variable which are directly solved. Because the equations in this example are linear and they can be totally uncoupled, the process looks trivial. Things can get a little more interesting if the blocks are a little less obvious. Consider the following example with 8 linear equations in 8 unknowns:

$$
\begin{array}{lllllllll}
x_1 & & & & & & & & = 11 \\
\end{array}
$$

$$
\begin{array}{llllllllr}
 & & x_3 & & & & & + x_8 & = 11 \\
 & & & & & & x_7 & & = 7 \\
 & & & & x_5 & - x_6 & - x_7 & & = -8 \\
x_1 & & & + x_4 & & - x_6 & & & = -1 \\
 & x_2 & & & & & & + x_8 & = 10 \\
 & & x_3 & & - x_5 & & & + x_8 & = 6 \\
 & & & x_4 & & & & & = 4 \\
x_1 & & & & & + x_6 & + x_7 & & = 14
\end{array}
$$

These equations and variables can be re-numbered and blocked. Each block is solved in turn. In the case above, blocking allows the equations to be solved in 6 blocks as follows:

Block 1: Equation 7

$x_4 = 4$

Block 2: Equation 2

$x_7 = 7$

Block 3: Equations 4 and 8

$x_1 + x_4 - x_6 = -1$ From here $x_1 = 1$

$x_1 + x_6 + x_7 = 14$ and: $x_6 = 6$

Block 4: Equation 3

$x_5 - x_6 - x_7 = -8$ From here: $x_5 = 5$

Block 5: Equations 1 and 6

$x_3 + x_8 = 11$ From here: $x_3 = 3$

$x_3 - x_5 + x_8 = 6$ and: $x_8 = 8$

Block 6: Equation 5:

$x_2 + x_8 = 10$ From here: $x_2 = 2$

The first two blocks contain a single equation with a single variable. These blocks simply define constants. EES will recognize that equations that depend from the start on a single variable are in reality parameter or constant definitions. These parameters are determined before any solution of the remaining equations takes place. No lower and upper limits on the guesses are needed for parameters, since the values of these parameters are determined immediately. The solution of the remaining equations is now very simple, although it did not appear trivial at the beginning of the process.

Grouping of equations is useful when the equations are linear, but it is not essential. When the equations are nonlinear, grouping of equations is nearly indispensable, otherwise later groups of equations begin iterating with totally incorrect values of earlier variables. The result is often divergence. EES is able to recognize groups of equations prior to solution by inspecting the Jacobian matrix using the Tarjan [8] algorithm. See reference [6] for more details on this algorithm.

Determination of Minimum or Maximum Values

EES has the capability to find the minimum or maximum (i.e., optimum) value of a variable when there is one to ten degrees of freedom (i.e., the number of variables minus the number of equations). For problems with a single degree of freedom, EES can use either of two basic algorithms to find a minimum or maximum: a recursive quadratic approximation known as Brent's method or a Golden Section search [9]. The user specifies the method, the variable to be optimized and an independent variable whose value will be manipulated between specified lower and upper bounds. When there are two or more degrees of freedom, EES uses Brent's method repeatedly to determine the minimum or maximum along a particular direction. The direction is determined by a direct search algorithm known as Powell's method or by the conjugate gradient method [9,10].

The recursive quadratic approximation algorithm proceeds by determining the value of the variable which is to be optimized for three different values of the independent variable. A quadratic function is fit through these three points. Then the quadratic function is differentiated analytically to locate an estimate of the extremum point. If the relationship between the variable which is being optimized and the independent variable is truly quadratic, the optimum is found directly. If this is not the case, the algorithm will use the newly obtained estimate of the optimum point and two (of the three) points which are closest to it to repeat the quadratic fit. This process is continued until the convergence criteria set for the minimization/maximization process are satisfied.

The Golden Section search method is a region-elimination method in which the lower and upper bounds for the independent variable specified by the user are moved closer to each other with each iteration. The region between the bounds is broken into two sections, as shown in Figure 3. The value of the dependent variable is determined in each section. The bounds for the section which contains the smaller (for minimization) or larger (for maximization) dependent variable replace the interval bounds for the next iteration. Each iteration reduces the distance between the two bounds by a factor of (1-τ) where τ =0.61803 is known as the Golden Section ratio.

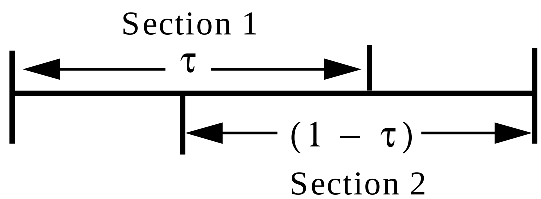


Figure 3: Region Elimination using the Golden Section Method

Numerical Integration

EES integrates functions and solves differential equations using a variant of the trapezoid rule along with a predictor-corrector algorithm. In explaining this method, it is helpful to compare the numerical scheme with the manner one would use to graphically determine the value of an integral.

Consider the problem of graphically estimating the integral of the function

$$f = 5 - 5 X + 10 X^2$$

for X between 0 and 1. In graphical integration, a plot of f versus X would be prepared. The abscissa of the plot would be divided into a number of sections as shown below. The area under the curve in each section is estimated as the area of a rectangle with its base equal to the width of the section and its height equal to the average ordinate value in the section. For example, the ordinate values at 0 and 0.2 in the plot below are 5 and 4.4, respectively. The area of the first section is then 0.2 * (5+4.4)/2 or 0.94. The estimate integral value between 0 and 1 is the sum of the areas of the 5 sections. The accuracy of this method improves as the number of sections is increased.

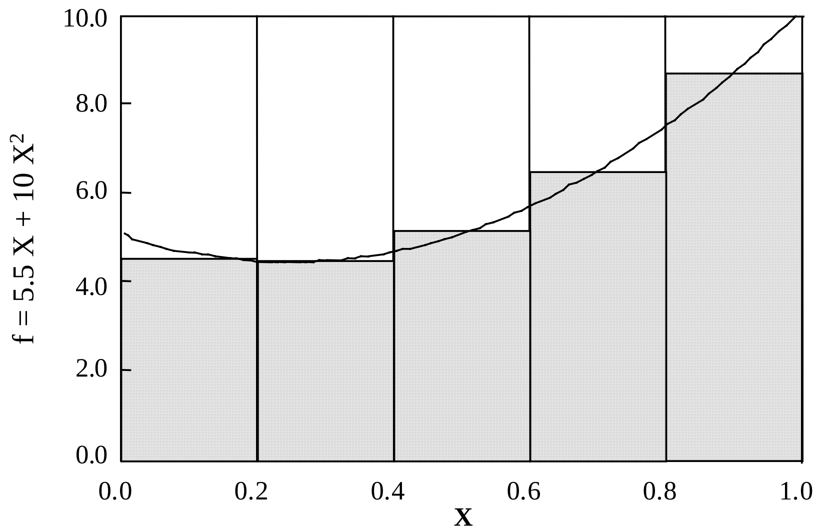


Figure 4: Numerical approximation of an integral

Integration in EES takes place in a manner quite analogous to the graphical representation. The abscissa variable, X, in the example above, is placed in the Parametric Table. The values of X entered into the table identify the width of each section. EES does not require each section to have the same width. The numerical value of the function, f, which is to be integrated, is evaluated at each value of X and supplied to EES through the Integrate function, e.g., Integral(f,X).

In some situations, such as in the solution of differential equations, the value of f may not be explicitly known at a particular value of X. The value of f may depend upon the solution to non-linear algebraic equations which have not yet converged. Further, the value of f may depend upon the value of the integral up to that point. In this case, iteration is needed. EES will repeatedly evaluate the section area using the latest estimate of f at the current value of X until convergence is obtained. The procedure in which the estimate of the integral made on the first calculation is corrected with later information is referred to as a predictor-corrector algorithm.

References

1. A. W. Al-Khafaji and J. R. Tooley, ***Numerical Methods in Engineering Practice***, Holt, Rinehart and Winston, 1986, pp. 190 & ff.

2. C. F. Gerald and P. O. Wheatley, ***Applied Numerical Analysis***, Addison-Wesley 1984, pp. 135 & ff.

3. J. H. Ferziger, ***Numerical Methods for Engineering Application***, Wiley-Interscience 1981, Appendix B.

4. F. S. Acton, ***Numerical Methods that Usually Work***, Harper and Row 1970.

5. I. S. Duff, A. M. Erisman and J. K. Reid, ***Direct Methods for Sparse Matrices***, 1986 Oxford Science Publications, Clarendon Press.

6. S. Pissanetsky, "Sparse Matrix Technology," Academic Press 1984.

7. F. L. Alvarado, "The Sparse Matrix Manipulation System," ***Report ECE-89-1***, Department of Electrical and Computer Engineering, The University of Wisconsin, Madison, Wisconsin, January 1989.

8. Tarjan, R. "Depth-First Search and Linear Graph Algorithms," ***SIAM J. Comput.*** 1, 146-160, (1972)

9. Powell's Method of Successive Quadratic Approximations, Reklaitis, Ravindran and Radsdell, ***Engineering Optimization***, John Wiley, New York, (1983)

10. W. H. Press, B. P. Flannery and S. A. Teukolsky, and Vetterling, W.T., ***Numerical Recipes in Pascal***, Cambridge University Press, Chapter 10, (1989)

Appendix C

Adding Property Data to EES

Background Information

EES uses an equation of state approach rather than internal tabular data to calculate the properties of fluids. For some substances and conditions, the ideal gas law is applicable. EES employs a naming convention to distinguish ideal gas and real fluid substances. Substances which are represented by their chemical symbol (e.g., N2) are modeled with the ideal gas law whereas substances for which the name is spelled out (e.g., Nitrogen) are considered to be real fluids. (Air and AirH2O are exceptions to this naming convention.)

Ideal gas substances rely on JANAF table data [Stull, 1971] to provide the enthalpy of formation and absolute entropy at a reference state of 298 K, 1 atm. Specific heat correlations for these gases and the ideal gas law are used to calculate the thermodynamic properties at conditions other than the reference state. A number of ideal gas substances are built into EES. The external JANAF program provides thermodynamic property information for hundreds of additional substances. Additional ideal gas fluid data can be added with .IDG files in the USERLIB folder, as explained below.

The Martin-Hou [1955] equation of state (or variations of it) is used for all real fluids except water. (Several equations of state are provided for water, the most accurate and computationally intensive being the equation of state published by Harr, Gallagher, and Kell [1984]. Ice properties rely upon correlations developed by Hyland and Wexler [1983].) Thermodynamic property relations are used to determine enthalpy, internal energy and entropy values based upon the equation of state and additional correlations for liquid density, vapor pressure, and zero-pressure specific heat as a function of temperature. A modification to the Martin-Hou equation of state proposed by Bivens et al. [1996] allows this equation equation of state to be applied for mixtures, such as the R400 refrigerant blends.

Viscosity and thermal conductivity of liquids and low-pressure gases are correlated as polynomials in temperature. Temperature alone determines the transport properties for ideal gases. For real fluids, the effect of pressure on the gas transport properties is estimated using correlations from Reid et al. [1977].

Adding Fluid Properties to EES

EES has been designed to allow additional (ideal and real) fluids to be added to the property data base. The user must supply the necessary parameters for the thermodynamic and transport property correlations. The parameters are placed in an ASCII text file which must be located in the EES\USERLIB subdirectory. EES will load all fluid files found in the EES\USERLIB subdirectory at startup. The additional fluids will appear in every way identical to the built-in fluids. The following sections describe the format required for the property data files.

<u>Ideal Gas files</u>

Ideal gas files must have a .IDG filename extension. An equation of state is not needed since it is assumed that the fluid obeys the ideal gas equation of state. However, particular attention must be paid to the reference states if the gas is to be used in calculations involving chemical reactions. The enthalpy of formation and Third-law entropy values at 298 K and 1 bar (or 1 atm) must be supplied. An example file providing the parameters for CO_2 is provided below. The properties of ideal gas fluid can be entered by adapting the file format to the new fluid.

SAMPLE TESTCO2.IDG File

```
TestCO2
44.01        {Molar mass of fluid
100.0        {Tn Normalizing value in K}
250          {Lower temperature limit of Cp correlation in K}
1500         {Upper temperature limit of Cp correlation in K}
-3.7357 0    {a0, b0  Cp=sum(a[i]*(T/Tn)^b[i], i=0,9 in kJ/kgmole-K}
30.529  0.5 {a1, b1}
-4.1034 1.0 {a2, b2}
0.02420 2.0 {a3, b3}
0 0          {a4, b4}
0 0          {a5, b5}
0 0          {a6, b6}
0 0          {a7, b7}
0 0          {a8, b8}
0 0          {a9, b9}
298.15       {TRef in K}
100          {Pref in kPa}
-393520      {hform - enthalpy of formation in kJ/kgmole at TRef}
213.685      {s0 - Third law entropy in kJ/kgmole-K at Tref and PRef}
0            {reserved - set to 0}
0            {reserved - set to 0}
200           {Lower temperature limit of gas phase viscosity correlation
in K}
1000          {Upper temperature limit of gas phase viscosity correlation
in K}
-8.09519E-7 {v0  Viscosity = sum(v[i]*T^(i-1)) for i=0 to 5 in Pa/m^2}
6.039533E-8 {v1}
-2.8249E-11 {v2}
9.84378E-15 {v3}
```

```
-1.4732E-18 {v4}
0           {v5}
200    {Lower  temperature  limit  of  gas  phase  thermal  conductivity
correlation in K}
1000  {Upper  temperature  limit  of  gas  phase  thermal  conductivity
correlation in K}
-1.1582E-3  {t0  Thermal Conductivity = sum(t[i]*T^(i-1)) for i=0 to 5
in W/m-K}
3.9174E-5   {t1}
8.2396E-8   {t2}
-5.3105E-11 {t3}
3.1368E-16  {t4}
0           {t5}
0           {Terminator - set to 0}
```

Real Fluid Files

A pure real fluid is identified with a .MHE (for Martin-Hou Equation) filename extension. A sample file named XFLUID.MHE is listed on the following pages illustrating the required file format. (The sample file contains the parameters used for n-butane.) The file consists of 75 lines. The first line provides the name of the fluid which EES will recognize in the property function statements. For example, the first line in the sample file contains UserFluid. The enthalpy for this substance would then be obtained as follows.

h=**Enthalpy**(UserFluid,T=T1, P=P1)

The fluid name will appear in alphabetical order with other fluid names in the Function Information dialog window. The following 74 lines each contain one number. A comment follows on the same line (after one or more spaces) to identify the number. The forms of all of the correlations except the pressure-volume-temperature relation are indicated in the XFLUID.MHE file. Pressure, volume and temperature are related by the Martin-Hou equation of state in the following form. A method for obtaining the coefficients is described by Martin and Hou, [1955].

Martin-Hou Equation of State (parameters in lines 18-36)

$$P = \frac{R\,T}{v-b} + \frac{A_2 + B_2 T + C_2 e^{-\beta T/T_c}}{(v-b)^2} + \frac{A_3 + B_3 T + C_3 e^{-\beta T/T_c}}{(v-b)^3} + \frac{A_4 + B_4 T + C_4 e^{-\beta T/T_c}}{(v-b)^4} + \frac{A_5 + B_5 T + C_5 e^{-\beta T/T_c}}{(v-b)^5} + \frac{A_6 + B_6 T + C_6 e^{-\beta T/T_c}}{e^{\alpha v}(1 + C' e^{\alpha v})}$$

where

P [=] psia, T [=] R, and v [=] ft^3/lb_m

You may need to curve fit tabular property data or data obtained form a correlation in a different form to obtain the appropriate parameters. Most of the correlations are linear with respect to the parameters so that they can be determined by linear regression. A parameter set which improves upon the fit resulting from the Martin and Hou method can be determined by non-linear regression. EES can be used to do these regressions.

SAMPLE XFLUID.MHE File for pure fluids

```
UserFluid
58.1            { molecular weight}
0               { not used}
12.84149        { a}  Liquid
Density=a+b*Tz^(1/3)+c*Tz^(2/3)+d*Tz+e*Tz^(4/3)+f*sqrt(Tz)+g*(Tz)^2}
33.02582        { b}     where Tz=(1-T/Tc) and Liquid
Density[=]lbm/ft3
-2.53317        { c}
-0.07982        { d}
9.89109         { e}
0               { f}
0               { g}
-6481.15338     { a}   Vapor pressure fit: lnP=a/T+b+cT+d(1-
T/Tc)^1.5+eT^2
15.31880        { b}      where T[=]R and P[=]psia
-0.0006874      { c}
4.28739         { d}
0               { e}
0               { not used}
0.184697        { Gas constant in psia-ft3/lbm-R}
1.5259e-2       { b}   Constants for Martin-Hou EOS/English_units
-20.589         { A2}
9.6163e-3       { B2}
-314.538        { C2}
0.935527        { A3}
-3.4550e-4      { B3}
19.0974         { C3}
-1.9478e-2      { A4}
0               { B4}
0               { C4}
0               { A5}
2.9368e-7       { B5}
-5.1463e-3      { C5}
0               { A6}
0               { B6}
0               { C6}
5.475           { Beta}
0               { alpha}
0               { C’}
-7.39053E-3     { a}    Cv(0 pressure) = a + b T + c T^2 + d T^3 +
e/T^2
6.4925e-4       { b}            where T[=]R and Cv[=]Btu/lb-R
9.0466e-8       { c}
-1.1273e-10     { d}
```

```
5.2005e3        { e}
124.19551       { href offset}
0.0956305       { sref offset}
550.6           { Pc [=] psia}
765.3           { Tc [=] R}
0.07064         { vc [=] ft3/lbm}
0               { not used}
0               { not used}
2               { Viscosity correlation type: set to 2: do not change}
260             { Lower limit of gas viscosity correlation in K}
535             { Upper limit of gas viscosity correlation in K}
-3.790619e6     { A}     GasViscosity*1E12=A+B*T+C*T^2+D*T^3
5.42356586e4    { B}     where T[=]K and GasViscosity[=]N-s/m2
-7.09216279e1   { C}
5.33070354e-2   { D}
115             { Lower limit of liquid viscosity correlation in K}
235             { Upper limit of liquid viscosity correlation in K}
2.79677345e3    { A}    Liquid Viscosity*1E6=A+B*T+C*T^2+D*T^3
-2.05162697e1   { B}     where T[=]K and Liquid Viscosity[=]N-s/m2
5.3698529e-2    { C}
-4.88512807e-5  { D}
2               { Conductivity correlation type: set to 2: do not
change}
250             { Lower limit of gas conductivity correlation in K}
535             { Upper limit of gas conductivity correlation in K}
7.5931e-3       { A}    GasConductivity=A+B*T+C*T^2+D*T^3
-6.3846e-5      { B}     where T[=]K and GasConductivity[=]W/m-K
3.95367e-7      { C}
-2.9508e-10     { D}
115             { Lower limit of liquid conductivity correlation in K}
235             { Upper limit of liquid conductivity correlation in K}
2.776919161e-1  { A}    LiquidConductivity=A+B*T+C*T^2+D*T^3
-8.45278149e-4  { B}     where T[=]K and LiquidConductivity[=]W/m-K
1.57860101e-6   { C}
-1.8381151e-9   { D}
0               { not used: terminator}
```

Fluid Properties for Blends

The Martin-Hou equation of state can be adapted for mixtures as proposed by Bivens et. al. The major modifications needed to make this pure component equation of state applicable to blends is to provide separate correlations for the bubble and dew point vapor pressures and a correlation for the enthalpy of vaporization, since the equation of state can not provide this information. Shown below is a listing of the R410A.MHE file that is used to provide property data for R410A, along with an explanation of each line in the file.

```
R410A
```

```
72.584          {molecular weight Bivens and Yokozeki}
400             {Indicator for blend}
30.5148         {a} Liquid density = a+b*Tz^(1/3)+c*Tz^(2/3)+d*Tz
60.5637         {b}       +e*Tz^(4/3)+f*sqrt(Tz)+g*(Tz)^2}
-5.39377        {c} where Tz=(1-T/Tc) and Liquid Density[=]lbm/ft3
55.5360815      {d}
-21.88425       {e}
0               {f}
0               {g}
-5.9789E+03  -5.9940E+03  {a} Bubble and Dew Pt Vapor pressure fit:
24.06932  24.04507        {b}     lnP=a/T+b+cT+d(1-T/Tc)^1.5+eT^2
-2.1192E-02  -2.1084E-02  {c}     where T[=]R and P[=]psia fit
-5.5841E-01  -4.4382E-01  {d}
1.3718E-05  1.3668E-05    {e}
0  0                      {not used}
0.1478          {Gas constant in psia-ft3/lbm-R}
0.006976        {b}  Constants for Martin-Hou EOS/English_units from Bivens
-6.40764E+00    {A2}
3.40372E-03     {B2}
-2.34220E+02    {C2}
1.41972E-01     {A3}
4.84456E-06     {B3}
9.13546E+00     {C3}
-4.13400E-03    {A4}
0               {B4}
0               {C4}
-9.54645E-05    {A5}
1.17310E-07     {B5}
2.45370E-02     {C5}
0               {A6}
0               {B6}
0               {C6}
5.75            {Beta}
0               {alpha}
0               {C'}
0.036582        {a}   Cv(0 pressure) = a + b T + c T^2 + d T^3 + e/T^2
2.808787E-4     {b}           where T[=]R and Cv[=]Btu/lb-R from Bivens
-7.264730E-8    {c}
2.6612670E-12   {d}
0               {e}
65.831547       {href offset}
-0.082942       {sref offset}
714.5           {Pc [=] psia}
621.5           {Tc [=] R}
0.03276         {vc [=] ft3/lbm}
0               {not used}
7               {# of coefficients which follow - used for blends}
1               {DeltaH Correlation type}
0.5541498       {Xo}
87.50197        {A} DeltaH_vap=A+B*X+C*X^2+D*X^3+E*X^4 Bivens
```

```
185.3407       {B}   where X =(1-T/Tc)^.333-X0, T in R and enthalpy in Btu/lb
13.75282       {C}
0              {D}
0              {E}
2              {Viscosity correlation type: set to 2: do not change}
200            {Lower limit of gas viscosity correlation in K}
500            {Upper limit of gas viscosity correlation in K}
-1.300419E6    {A}    GasViscosity*1E12=A+B*T+C*T^2+D*T^3
5.39552e4      {B}    where T[=]K and GasViscosity[=]N-s/m2
-1.550729e1    {C}
0              {D}
-999           {Lower limit of liquid viscosity correlation in K}
-999           {Upper limit of liquid viscosity correlation in K}
0              {A}   Liquid Viscosity*1E6=A+B*T+C*T^2+D*T^3
0              {B}    where T[=]K and Liquid Viscosity[=]N-s/m2
0              {C}
0              {D}
2              {Conductivity correlation type: set to 2: do not change}
200            {Lower limit of gas conductivity correlation in K}
500            {Upper limit of gas conductivity correlation in K}
-8.643088e-3   {A}   GasConductivity=A+B*T+C*T^2+D*T^3
7.652083e-5    {B}    where T[=]K and GasConductivity[=]W/m-K
2.144608e-9    {C}
0              {D}
-999           {Lower limit of liquid conductivity correlation in K}
-999           {Upper limit of liquid conductivity correlation in K}
0              {A}   LiquidConductivity=A+B*T+C*T^2+D*T^3
0              {B}    where T[=]K and LiquidConductivity[=]W/m-K
0              {C}
0              {D}
0 {terminator}
{The forms of the correlations and in some cases the coefficients have been
adapted from D.B. Bivens and A. Yokozeki, "Thermodynamics and Performance
Potential of R-410a," 1996 Intl. Conference on Ozone Protection Technologies
Oct, 21-23, Washington, DC.}
```

References

ASHRAE Handbook of Fundamentals, (1989, 1993, 1997), American Society of Heating, Refrigerating and Air Conditioning Engineers, Atlanta, GA

ASHRAE, **Thermophysical Properties of Refrigerants**, American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Atlanta, GA, (1976)

D.B. Bivens and A. Yokozeki, "Thermodynamics and Performance Potential of R-410a," 1996 Intl. Conference on Ozone Protection Technologies Oct, 21-23, Washington, DC.

Downing, R.C. and Knight, B.W., "Computer Program for Calculating Properties for the "FREON" Refrigerants," DuPont Technical Bulletin RT-52, (1971); Downing, R.C., "Refrigerant Equations", ASHRAE Transactions, Paper No. 2313, Vol. 80, pt.2, pp. 158-169, (1974)

Gallagher, J., McLinden, M, Morrison, G., and Huber, M., REFPROP - NIST Thermodynamic Properties of Refrigerants and Refrigerant Mixtures, Versions 4, 5, and 6, NIST Standard Reference Database 23, NIST, Gaithersburg. MD 20899, (1989)

Harr, L. Gallagher, J.S., and Kell, G.S (Hemisphere, 1984). **NBS/NRC Steam Tables**, Hemisphere Publishing Company, Washington, (1984)

Howell, J.R., and Buckius, R.O., **Fundamentals of Engineering Thermodynamics**, McGraw-Hill, New York, (1987)

Hyland and Wexler, “Formulations for the Thermodynamic Properties of the Saturated Phases of H_2O from 173.15 K to 473.15 K, ASHRAE Transactions, Part 2A,Paper 2793 (RP-216), (1983)

Keenan, J.H., Chao, J., and Kaye, J., **Gas Tables**, Second Edition, John Wiley, New York, (1980)

Keenan, J.H. et al., **Steam Tables**, John Wiley, New York, (1969)

Irvine, T.F. Jr., and Liley, P.E., **Steam and Gas Tables with Computer Equations**, Academic Press Inc., (1984)

Martin, J.J. and Hou, Y.C., ”Development of an Equation of State for Gases,” A.I.Ch.E Journal, 1:142, (1955)

McLinden, M.O. et al., "Measurement and Formulation of the Thermodynamic Properties of Refrigerants 134a and 123, ASHRAE Trans., Vol. 95, No. 2, (1989)

Reid, R.C.Prausnitz, J.M. and Sherwood, T.K., **The Properties of Gases and Liquids**, McGraw-Hill, 3rd edition, (1977)

Shankland, I.R., Basu, R.S., and Wilson, D.P., "Thermal Conductivity and Viscosity of a New Stratospherically Sate Refrigerant - 1,1,1,2 Tetrafluoroethane (R-134a), published in **CFCs:**

Time of Transition, American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc., (1989)

Shankland, I.R., "Transport Properties of CFC Alternatives", AIChE Spring Meeting, Symposium on Global Climate Change and Refrigerant Properties, Orlando, FL, March, (1990)

Stull, D.R., and Prophet, H., **JANAF Thermochemical Tables**, Second Edition, U.S. National Bureau of Standards, Washington, (1971)

Van Wylen, G.J., and Sonntag, R.E., **Fundamentals of Classical Thermodynamics**, Third Edition, John Wiley, New York, (1986)

Wilson, D.P. and Basu, R.S., "Thermodynamic Properties of a New Stratospherically Safe Working Fluid - Refrigerant 134a", paper presented at the ASHRAE meeting, Ottawa, Ontario, Canada, June, (1988), published in **CFCs: Time of Transition**, American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc., (1989)

Appendix D

Example Problem Information

The EXAMPLES subdirectory within the EES directory contains many worked-out example problems. Each example problem illustrates one or more EES features, as indicated in the information below.

Feature	**EES Example Files**
Arrays	MATRIX.EES, MATRIX2.EES, RANKINE.EES, REFRIG.EES, REGEN.EES
Complex numbers	COMPLEXROOTS.EES
Comments	HEATEX.EES
Curve-fitting	COPPER.EES
Diagram window	DIAGRMW.EES
Differential equations	DRAG.EES, RK4_TEST.EES, SUBSTEPS.EES, DIFEQN1.EES, DIFEQN1.EES
Differentiate function	COPPER.EES
DUPLICATE command	MATRIX.EES, MATRIX2.EES, NLINRG.EES, REGEN.EES
Formatted Equations	HEATEX.EES, DRAG.EES
Functions, user-written	CONVECT.EES, MOODY.EES, RK4_TEST.EES
Greek symbols	HEATEX.EES, NLINRG.EES
Integration	DBL_INTEG.EES, DIFEQN1.EES, DIFEQN2.EES, SUBSTEPS.EES, DRAG.EES, RK4_TEST.EES
Interpolate function	COPPER.EES
JANAF table	FLAMET.EES
LOOKUP table	NLINRG.EES, COPPER.EES
Minimize or maximize	MAXPOWER.EES, NLINRG.EES, RANKINE.EES
Modules	MOODY.EES
Overlay Plot	CH1EX.EES RANKINE.EES
Parametric table	CAPVST.EES, CH1EX.EES, DIFEQN1.EES, FLAMET.EES, SUBSTEPS.EES
Plotting	CAPVST.EES, DIFEQN2
Procedures, user-written	REGEN.EES
Procedures, external	ABSORP.EES
Properties, thermodynamic	REFRIG.EES, CATVST.EES, CH1EX.EES, FLAMET.EES, REGEN.EES
Property Plot	RANKINE.EES, REFRIG.EES
Psychrometric functions	SUPERMKT.EES
Regression	NLINRG.EES
Subscripted variables	MATRIX.EES, MATRIX2.EES, HEATEX.EES
SUM function	MATRIX.EES, MATRIX2.EES, NLINRG.EES
Systems of equations	HEATEX.EES, CH1EX.EES
TABLEVALUE	DIFEQN2.EES
Transport properties	CONVECT.EES
Unit conversion	DRAG.EES